

**WO 02/087157 A1**

**WO 02/087157 A1**

---



*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

## **GRAPHICAL USER INTERFACE FOR A TRANSPORT MULTIPLEXER**

### **CROSS REFERENCE TO RELATED CASES**

**[0001]** This application claims the benefit under 35 U.S.C. 119(e) of co-pending U.S. Provisional Application Nos. 60/285,188 filed April 20, 2001 and entitled "Broadband Bandwidth Management, Device Management and Multi-Media Control System"; and 60/285,153 filed April 20, 2001 and entitled "Data Insertion, Transport, Grooming, Aliasing, Routing, and Multiplexing of MPEG 2 Data Streams", which Provisional Applications are hereby incorporated by reference.

**[0002]** This application is related to co-pending U.S. Patent Application Serial No. \_\_/\_\_, filed April 16, 2002 (based on provisional application no. 60/322,063 filed September 13, 2001) and entitled "High Speed Serial Data Transport Between Communications Hardware Modules," which Application is hereby incorporated by reference.

### **BACKGROUND OF THE INVENTION**

#### **1. Field of the Invention**

**[0003]** The present invention is directed to systems, processes, methodologies, apparatus and related software to provide graphical user interfaces for use with communications hardware. More particularly, the invention relates to the field of graphical user interfaces that facilitate monitoring and configuration of content stream routing through transport multiplexers. Accordingly, the general objects of the invention are to provide novel systems, methods, apparatus and software of such character.

#### **2. Description of the Related Art**

**[0004]** Broadband media convergence between video, audio and data creates a chaotic environment of different standards and legacy communications technologies. The relationship between physical and logical resources in such systems needs to be manipulated and communicated between broadband hardware, its control system and the operator. Conventionally, this is accomplished with the use of dedicated control computers for each piece of communications hardware associated with the task of routing multimedia content. Therefore, modern communications hardware personnel, such as cable operators and television programmers, use great numbers of diverse equipment, each of which is managed by a separate control system hosted on a separate computer. Conventional communications systems of this nature also typically require the use of hyper-technical operator interfaces in order to configure conventional broadband equipment. For these and other reasons, most of such systems are nearly intolerably difficult to utilize, counter-intuitive, cryptic, inefficient and expensive to purchase and operate.

**[0005]** Another problem with conventional broadband communications hardware is its inability to conveniently provide operators with information regarding the system hardware and software. This creates many inefficiencies in operation of such equipment. As but one example, troubleshooting system errors is currently a difficult and expensive process because system operators must physically inspect the broadband communications hardware in order to determine the system hardware utilized and its operational status. In particular, troubleshooting system difficulties may entail operator inspections of a communications hardware rack to determine if all of the communications hardware has been plugged-in, turned on, connected to a source of content stream and/or operating in a particular way.

**[0006]** In addition to the difficulties associated with determining hardware configuration and status, conventional broadband communications hardware makes it difficult to determine current content stream routing configurations. This is, in part, due to the fact that current broadband communications hardware is not sophisticated enough to indicate various attributes of incoming data streams. Additionally, current user interfaces

for conventional broadband communications hardware require highly skilled personnel to configure routing control for various programs. The difficulties associated with conventional methods of routing control make it extremely difficult to efficiently utilize system bandwidth. Therefore, operators of conventional systems of this nature often underutilize the bandwidth available to perform content stream routing. Given that transport bandwidth is still a precious commodity in this field, this is a significant deficiency.

[0007] A further deficiency associated with conventional broadband communications hardware is that such hardware must be manually configured. Thus, current systems typically require operators to be skilled in the use of a cryptic and sophisticated computer code in order to operate conventional broadband communications hardware. As the demand for information/content options continues to increase, it becomes increasingly difficult to manually configure broadband communications hardware to route content streams. Moreover, such systems must be continually operated and appropriately configured each time a content stream routing change is to be executed.

[0008] There is, accordingly, a need in the art for novel methods, systems and apparatus that provide a simple and real-time display of content stream routing through a transport multiplexer. Such methods and apparatus should include an intuitive graphical user interface that supplies the operator with status information regarding associated system hardware and software. Such methods and apparatus should also include a feature-rich graphical user interface that enables an operator to easily monitor and manipulate content streams flowing through a transport multiplexer.

[0009] There is, accordingly, an additional need in the art for novel methods, systems and apparatus that permit configuration of content stream routing of plural stream components simultaneously. Such methods and apparatus should be flexible enough to permit operators to configure routing control of individual content stream components as well as groups of such components (e.g., programs, components, etc.) simultaneously.

**[0010]** There is, a further need in the art for novel methods, systems and apparatus that permit configuration of routing control of future routing events in a broadband multiplexer. Such methods and apparatus should also permit operators to configure transport multiplexers to automatically change routing configurations at a future point in time, even in the absence of the operator. Such a system would permit automated control of broadband media hardware by predetermining routing configuration information for extended time periods and enable the automatic execution of configuration changes.

**[0011]** There is still another need in the art for transport multiplexer graphical user interfaces that have the ability to display intuitive and consistent mnemonic identifiers for the various content streams that are displayed on various screens throughout the graphical user interface.

**[0012]** There is, accordingly, a further need in the art for novel methods, systems and apparatus that provides both manual and computer assisted configuration of content stream routing control through transport multiplexers. Such methods and apparatus would also preferably include a graphical user interface that permits the operator to configure routing control through simple manipulation of graphical objects such as icons.

**[0013]** There is, yet another need in the art for novel methods, systems and apparatus that provide real-time display of hardware identity/configuration and/or status data in a broadband multiplexer. Such methods and apparatus should enable system operators to identify the physical structure and configuration of communications hardware from a single graphical user interface, and thereby avoid the necessity of physically inspecting the communications hardware. Such methods and apparatus should also be capable of reflecting the current status of the communications hardware so that an operator may verify whether or not the various equipment is operating as expected.

## SUMMARY OF THE INVENTION

**[0014]** The present invention satisfies the above-stated needs and overcomes the above-stated and other deficiencies of the related art by providing methods, systems and apparatus that enables real-time monitoring and manipulation of content streams flowing through a transport multiplexer. The invention can take the form of a graphical user interface capable of displaying system hardware identity, structure, configuration and/or status information. The interface can also provide computer assisted routing configuration through simple manipulation of graphical objects such as drag and drop operations performed on icons. To further simplify operation, the invention has the ability to display intuitive and consistent mnemonic identifiers for the various content streams that are displayed on various screens throughout the graphical user interface. Routing control is further simplified by permitting operators to configure routing control of individual content stream components as well as groups of such components simultaneously. Still further flexibility is provided by configuration of future routing events, thereby enabling the automatic execution of configuration changes at a future time.

**[0015]** The invention can also take the form of a method of using a graphical user interface to configure routing control of a content stream through a transport multiplexer that has plural input and output ports. The inventive method comprises displaying icons representing the input and output ports and an icon representing an input content stream received by one of the input ports. The input stream icon is the dragged and dropped near the output port icon which causes display of an icon representing an output stream exiting the output port. The dragged and dropped output stream icon could represent an output stream presently exiting the output port or one that will begin exiting the output port at a predetermined time in the future. In an alternative method embodiment, the method can display an alphanumeric character string which can

be dragging and dropped with the input stream icon to thereby result in display of an alphanumeric character string that represents the output stream.

**[0016]** Naturally, the above-described methods of the invention are particularly well adapted for use with the above-described apparatus of the invention. Similarly, the apparatus of the invention are well suited to perform the inventive methods described above.

**[0017]** Numerous other advantages and features of the present invention will become apparent to those of ordinary skill in the art from the following detailed description of the preferred embodiments, from the claims and from the accompanying drawings.



## BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The preferred embodiments of the present invention will hereinafter be described in conjunction with the appended drawing figures, wherein like numerals denote like elements, and wherein:

[0019] Figure 1a illustrates the hardware architecture of a transport multiplexer in accordance with one preferred embodiment of the present invention;

[0020] Figure 1b illustrates a preferred form of the firmware host processor architecture of the transport multiplexer of Figure 1a;

[0021] Figure 2 illustrates system initialization and resource discovery processes for the transport multiplexer of Figure 1, the processes being in accordance with one preferred embodiment of the present invention;

[0022] Figure 3 illustrates various hardware attribute viewing capabilities in accordance with one preferred embodiment of the present invention;

[0023] Figure 4 illustrates system attribute viewing capabilities in accordance with one preferred embodiment of the present invention;

[0024] Figure 5 illustrates various output port enabling processes in accordance with one preferred embodiment of the present invention;

[0025] Figure 6 illustrates specification of present video and/or audio stream routing event(s) in accordance with one preferred embodiment of the present invention;

[0026] Figure 7 illustrates various system bandwidth utilization viewing capabilities in accordance with one preferred embodiment of the present invention;

[0027] Figure 8 illustrates certain event logging and viewing capabilities and processes in accordance with one preferred embodiment of the present invention;

[0028] Figure 9 illustrates specification of future content stream routing event(s) in accordance with one preferred embodiment of the present invention;

**[0029]** Figure 10 illustrates various IP data encapsulation and insertion capabilities in accordance with one preferred embodiment of the present invention; and

**[0030]** Figure 11 is a detailed flow chart illustrating the IP data encapsulation and insertion capabilities of Figure 10 in greater detail.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

**[0031]** The ensuing detailed description provides preferred exemplary embodiments only, and is not intended to limit the scope, applicability, or configuration of the invention. Rather, the ensuing detailed description of the preferred exemplary embodiments will provide those skilled in the art with an enabling description for implementing a preferred exemplary embodiment of the invention. It being understood that various changes may be made in the functional arrangement of elements without departing from the spirit and scope of the invention as set forth in the appended claims.

**[0032]** As shown in Figures 1a and 1b, TMX 20 includes a plurality of hardware, firmware and software components. Figure 1a is a hardware architecture diagram showing a representative transport multiplexer (TMX) 20 in accordance with one preferred embodiment of the present invention. As shown therein, the transport multiplexer can include, for example, a computer 22' (with an element manager 22 and a GUI 80) that is communicatively linked to a TMX chassis 42 via a network 24. TMX chassis 42 preferably includes a host processor board 40' (preferably with a VxWorks operating system), an input processor board 50', and output processor board 55', a multiplexer board 60' and a transcoding board 70'. These components are preferably, but not necessarily communicatively linked to one another with a single TMX chassis 42. The basic physical model of the preferred TMX chassis hardware is as follows: the TMX chassis is a container for boards, each board is a container for ports and processors, each processor is a container for firmware, and each port is a container for a protocol hierarchy (i.e., DS3, MPEG, TCP/IP, etc.). The chassis and each board have a set of state and status variables associated with them. These include: 1) an administrative state which is used to synchronize configuration access of multiple managers; 2) an operational state which is used to indicate whether or not the TMX (or a component of it) is in a fully enabled and operational state; and 3) an alarm status which is used to signal a variety of

alarm conditions by the TMX or a component thereof. As described in detail below, the host processor 40 controls the various hardware and software components of TMX 20 and stores MIB table data in accordance with SNMP for use by the various other components of the TMX and the element manager.

[0033] Transport multiplexer 20 is suited to a wide variety of application environments including: (1) cable headend; (2) satellite uplink; and (3) terrestrial broadcast. Communication between element manager 22 and TMX chassis 42 is preferably performed in accordance with a modified Simple Network Management Protocol (SNMP) and the content streams to be routed through transport multiplexer 20 are preferably in accordance with one of the well-known MPEG standards. Most preferably, the content streams are MPEG2 data streams. While some preferred embodiments of the present invention use some conventional MIB tables in accordance with well known SNMP standards, many of the MIB's referenced herein comprise novel data structures. These data structures are fully defined in the computer program listing provided at the end of this specification. Therefore, those of ordinary skill will more than amply understand the nature and function of those novel data structures based on the teachings contained herein.

[0034] A more detailed description of the hardware components of TMX 20 is provided in previously mentioned co-pending U.S. Patent Application, Serial No. \_\_\_\_/\_\_\_\_, filed April 16, 2002 and entitled "High Speed Serial Data Transport Between Communications Hardware Modules," which Application has been incorporated herein by reference. Accordingly, further detailed discussion of these hardware components is not necessary, a complete understanding of these components being achieved with reference to these incorporated applications.

The relationship between physical and logical resources in the system needs to be manipulated and communicated between TMX chassis 42, element manager 22 and human operators 10 (e.g., Figure 2). This is generally achieved by modeling the various hardware components of the system. The TMX modeling elements are integrated into the SNMP Management Information Base (MIBs) by using the standard "MIB 2" interfaces

table (ifTable) to represent each one of the modeling elements as an interface with specific extensions as specified below. This permits identification of any board and/or any port, by its table index: the ifindex in the ifTable.

**[0035]** With continuing reference to Figure 1a, element manager 22 is preferably linked to TMX chassis 42 by an Ethernet. It will be appreciated by those skilled in the art that other network technologies can alternatively be used. Element manager 22 may be uploaded as a java-application from TMX 42 to a remote computer using a browser, the remote computer preferably being communicatively linked to transport multiplexer 20 via network 24 during an initial set-up phase. Subsequently, element manager 22 can be run on the remote computer as a java program. The remote computer is preferably a conventional personal computer with a conventional operating system and browser, the system permitting control over TMX chassis 42 subsequent to installation of element manager 22. A graphical user interface (GUI) is preferably incorporated into element manager 22 and is described in detail below. The GUI is preferably presented to an operator on a conventional personal computer monitor (e.g., an LCD screen or a CRT monitor). A wide variety of alternative software and hardware components for hosting and operating graphical user interface and element manager 22 will readily occur to those of ordinary skill in the art based on the disclosure contained herein

**[0036]** Figure 1b illustrates various firmware and software components 52-69 of TMX 20 which are communicatively linked to one another as shown therein. These components include an SNMP agent 44, a message handler 45 and a fault manager 59. TMX 20 further comprises a resource manager 52, a configuration manager 46, a PAT/PMT collection module 54, a PSIP collection module 57, an input module 50, an IP encapsulation module 66, a time table manager 67 and a number of DSP API's. These include multiplexer processing 60, transcode processing 62 and quantization level processing 64. There is a one-to-one correspondence between these firmware modules and certain hardware components of the preferred embodiment. The corresponding hardware components can be found in Figure 1a and include input processor board 50',

multiplexer board 60' (with a multiplex processor 60 and a quantization level processor 61), a transcoding board 70' (with either 5 transcode processors 71 or 3 transcode processors, 1 multiplex processor and 1 QLP 71'). Consequently, when the configuration manager performs operations on the firmware modules, the corresponding hardware modules are also affected. The flow of information and commands between the various components within TMX chassis 42 is generally indicated in Figures 1a and 1b by the use of arrows. In particular, the flow of commands and information from element manager 22 is through SNMP agent 44, which translates SNMP protocol commands from element manager 22 into a conventional form so that they can be understood by the various other components of TMX 20. The preferred conventional communication protocol is a simple protocol in which a number indicative of a request or command is passed along with an associated data structure for receiving data to be manipulated in accordance with the associated command. Thus, SNMP agent 44 generally acts as a communication broker between element manager 22 and the host processor firmware. SNMP agent 44 allows SNMP based management of and control over firmware functionality such as grooming, splicing, data insertion, etc., because it provides an interface with the various firmware modules (e.g., input processing task 50, multiplexer processing 60, transcode processing 62 and quantization level processing 64) that ultimately provide the desired functionality.

[0037] Configuration manager 46 receives commands and information from SNMP agent 44 via MIB message handler 45 and determines how to utilize the hardware and other firmware to execute those commands at the card level. A detailed understanding of the various other components of TMX 20 will be obtained with reference to Figures 2 through 9 and the corresponding detailed description of these figures in the remainder of this specification.

[0038] Figures 2 through 11 illustrate the nine primary operational aspects of transport multiplexer 20. These nine operational aspects include (1) initialization and discovery of system resources 100; (2) view system hardware attributes 134; (3) view system software attributes 156; (4) enable output port 166; (5) specify present video and/or audio routing event(s) 184; (6) view bandwidth utilization 206; (7) view log

activity 222; (8) specify future routing event(s) 238; and (9) IP data encapsulation and insertion 260. These aspects of the present invention are discussed in detail immediately below.

**[0039]** With reference to Figure 2, there is illustrated therein system initialization and resource discovery processes for the broadband multiplexer of Figure 1, the processes being in accordance with one preferred embodiment of the present invention. As shown, initialization and discovery of the inventive system commences with power-up 101 of TMX chassis 42, whereupon resource manager 52 conducts discovery (at 102) of the hardware and system software information. Thus, TMX chassis 42 executes a number of functions at 104 to identify system components installed in TMX chassis 42. Also at 102, MIB message handler 45 populates the appropriate MIB's (ifTable and ifStack) with information and SNMP agent 44 awaits queries at 106. Upon completion of these tasks, TMX chassis 42 is prepared to execute various activities based on operator-driven commands delivered to TMX chassis 42 via element manager 22.

**[0040]** At this point, an operator 10 can start up element manager 22 in response to which the element manager, at 104, displays graphical user interface 80 showing a blank tree view screen 81 for viewing. Blank tree view screen 81 includes an input tree window 82, an output tree window 82' and a log message window 87. At 108, element manager 22 automatically reads the appropriate MIB's to discover the hardware that is currently installed in TMX 20. This includes system hardware attribute data such as port data and/or physical structure. There are several types of ports (e.g., ASI, DHEI, SMPTE 310, DS3) which are supported by the preferred embodiment of the present invention. Data for various port parameters is described /defined by the ifentry MIB table. At 110, element manager 22 downloads the appropriate DSP code to the IdentxTable MIB. SNMP agent 44 of TMX chassis 42 creates a new MIB entry at 112 and message handler 45 passes this information to configuration manager 46 for fulfillment. At this point, element manager 22 requests PAT data at 114. This request is processed by the TMX at 116. At 120, the PAT is parsed by element manager 22 so that the appropriate PMT's can be identified. These are requested at 122 and this request is

processed by the TMX at 124. After the requested information is generated, SNMP agent 44, awaits further queries at 126. This data is then read by element manager 22 at 128 and graphical user interface 80 is updated. In particular, the requested data is used to populate tree view screen 81 with system hardware icons 84 and 84' and, preferably mnemonic, hardware names 83 and 83' extracted from the data streams themselves using PSIP collection module 57. Operator 10 is, thus, presented with a visual representation of the system hardware components.

[0041] After receiving the system hardware attributes data from TMX chassis 42, element manager 22 proceeds to retrieve and display log messages that may have been generated at 130. This is achieved with the assistance of a fault manager 59 and SNMP agent 44 at 132. Thus, once log polling has commenced, element manager 22 displays the port and log data at 132 to graphical user interface 80 where the tree view screen is updated to display input ports 85, output ports 85' and log messages 88 in log message window 87. As shown, input and output ports 85 and 85' preferably have associated mnemonic and alphanumeric identifiers. The ports are also preferably color coded to indicate whether or not the ports are active. Upon reviewing the newly completed tree view screen 81, operator 10 can initiate various activities as described below with respect to Figures 3 through 10. These activities can include, for example, view system hardware attributes 134, view system software attributes 156, enable output port 166, specify present video and/or audio routing events 184, view bandwidth utilization 206, view log activity 222, specify future routing events 238 and IP data encapsulation and insertion event(s) 260. Various other related activities that can be performed by operator 10 will readily occur to those of ordinary skill in the art based on the disclosure contained herein.

[0042] Turning now to Figure 3, this Figure illustrates various system hardware attribute viewing processes 134 in accordance with one preferred embodiment of the present invention. The hardware processes shown in Figure 3 are initiated by operator 10 upon selection of the chassis view screen from the menu items at the top of tree view screen 81. This option is accessed by selecting the "view" menu item at the top



of the screen and selecting the chassis view option. Available hardware viewing options include "front chassis view" and "rear chassis view" and "system information." Upon selection of one of the chassis view options at the graphical user interface, element manager 22 gathers the requested hardware information from the appropriate MIB's (136) with the assistance of TMX chassis 42. This MIB data is provided by TMX chassis 42 as indicated by 138 and element manager 22 then displays the information on one of chassis view screen 89 and 90.

[0043] With continuing reference to Figure 3, one can see that graphical user interface 80 uses the received hardware and status data to display system hardware attributes and, in particular, chassis view screens 89 and 90 as initially requested by operator 10. Front chassis view screen 89 includes various graphical objects indicative of the identity of, physical structure of, configuration of and status of the various cards received within TMX chassis 42. In this illustrative example, these cards include CPU card 40", multiplexer card 60", first input processor board 50" and second input processor board 50". While it is also possible to receive log messages within log message window 87 of front chassis view screen 89, no log messages have been generated in this illustrative example.

[0044] Rear chassis view screen 90 can also be selected by operator 10 as an alternative to front chassis view screen 89. In this illustrative example, rear chassis view screen 90 includes various graphical objects indicative of the identity of, physical structure of, configuration of and status of the rear portion of the various cards received within TMX chassis 42 and discussed above with respect to the front chassis view. The log messages can, optionally, also be displayed in log message window 87 of rear chassis view screen 90. This aspect of the present invention allows an operator 10 to easily select, and then, view system hardware attributes in the manner discussed above. This feature of the present invention is particularly advantageous in that it allows an operator to troubleshoot difficulties with transport multiplexer 20 without having to physically access the communications hardware itself.

**[0045]** The preferred continuous hardware status polling features of the present invention are shown at 139. In particular, the LED status information provided in the chassis view screens is updated at regular intervals by the repeated execution of the functions shown in blocks 140 – 146.

**[0046]** Turning now to Figure 4, this figure illustrates system attribute viewing processes and capabilities 156 in accordance with one preferred embodiment of the present invention. As shown therein, viewing of system attributes such as board type, DSP attributes, software version, etc. commences with the initial system discovery process when the TMX executes the functions shown at 158. Thus, this information is readily available for display and SNMP agent 160 waits for such queries at 160. Upon selection of the version view menu option within the top portion of tree view screen 81 by operator 10, element manager 22 gathers the requested information at 162 displays it in system attributes screen 91. The data can then be viewed by operator 10 as desired. As shown in Figure 4 and Table 1 below, system attributes data displayed on screen 91 preferably includes the following data fields for the board and software running on each chassis slot:

<b>Table 1</b>
Board Name
TMX Application
JVM Version
System Name
IP Address
Chassis ID
Board Revision
FPGA Version
VxWorks 08
CPU Version
MAP Lib Version
MUX Version
QLP Version
TPE Version

[0047] In the illustrative embodiment of Figure 4, TMX chassis 42 is a mid-plane TMX chassis with five board slots in each half of the chassis. Accordingly, this illustrative example includes ten slots (five slots for each half-plane). A detailed description of the structure and operation of TMX chassis 42 is contained in the application incorporated by reference and a wide variety of variant arrangements will readily occur to those of skill in the art based on the disclosure contained herein.

[0048] As shown in Figures 3 and 4, the preferred embodiment of the present system includes a GUI with a system information tab with which an operator can access information about the system such as system name, system description, system up-time and system location. This feature of present invention operates in a manner that is generally analogous to the view software version feature shown in Figure 4 and described in connection therewith immediately above

[0049] Figure 5 illustrates various output port enabling capabilities in accordance with one preferred embodiment of the present invention. As shown therein, output port enabling is initiated upon selection by operator 10 of the particular port to be enabled. Upon selection of a port, element manager 22, at 168, displays the transport editor 92 with default values. Operator 10 can then view the default data and edit the data if desired, such as by changing the status from disabled to enabled. For example, an operator will typically enable a transport stream, name that stream and assign an information transfer bit rate for the selected port. Upon selection of the "OK" button, the transport editor is closed, and element manager 22 gathers transport information from the editor and places it in the appropriate MIB tables (see 170). The TMX chassis also uses this information to execute the enable request as indicated at 172. The MIB table could be either one of two types: TMXiftable (for most ports) or the TMXgiexttable (for DS3 ports) due to the varying information requirements of the different port types.

[0050] Further, element manager 22 creates a PAT at 174 and the PAT is output by the TMX as indicated at 176. Finally, the tree view screen 81 of the GUI is updated by the element manager as indicated at 178. Graphical user interface 80

indicates successful enablement of the desired port by changing the attributes of the port icons in tree view screen 81. This is preferably accomplished by changing the color of the port icons, but other alternatives (such as changes in shape, movement, location, size, sound, etc.) will readily occur to those of ordinary skill in the art. Operator 10 can, thus, visually confirm that port enablement was successfully completed by viewing the newly-updated graphical user interface 80.

**[0051]** Figure 6 illustrates various system-assisted video and/or audio routing capabilities in accordance with one preferred embodiment of the present invention. As described in greater detail below, the present invention enables operator 10 to define and execute content stream routing either manually or semi-automatically. In particular, the preferred embodiment of the present invention provides operators with the ability to manually enter routing data element by element or, alternatively, to drag and drop graphical objects to and from various locations of the tree view screen 81. Element manager 22 cooperates with graphical user interface 80 to execute the various routing specification commands specified by corresponding drag and drop operations. This is achieved with automated population of MIB tables corresponding to the various actual fields necessary to define a routing command. Drag and drop operations on graphical user interface 80 assist operator 10 in defining video, audio and/or IP data routing events for the system. Defining routing specifications in this way is, therefore, semi-automatic.

**[0052]** Drag and drop operations on the graphical user interface can be used to perform a variety of related content stream routing functions. These include the ability to drag different levels from the input tree to the output tree. For example, an operator may drag (1) the content streams of an entire input port (possibly including plural programs, each of which possibly includes plural components) to an output port; (2) a complete program of an input port to an output port; (3) a complete program from an input port to a program of an output port; and (4) a component from an input port to an output port. A number of other drag and drop features will readily occur to those of ordinary skill in the art based on the disclosure contained herein. However, it should be noted that this portion of the specification specifically addresses content stream routing

that occurs in the present. The invention, however, also envisions configuration of content stream routing to be automatically executed at a future time (see, e.g., Figure 9). As described in greater detail below, content stream routing processes described immediately below (applicable to execution of present routing commands) are compatible with, and form a portion of, routing processes for execution of routing events in the future.

[0053] With primary reference to Figure 6, operator 10 can specify one or more present routing events by selecting the graphical objects representing one or more content streams to be routed to a desired location (e.g., an output port). The content stream could be either simple or contain plural components which may or may not be related to one another in one or more ways. For example, the object may represent a single component content stream, plural content streams that collectively constitute a program, or plural content streams that collectively constitute data streams present on an entire input port. In the illustrative example discussed immediately below, operator 10 drags the content streams for an entire port from the input tree to the output tree and proceeds to edit video and audio components of one program from the port.

[0054] Assisted routing in accordance with the invention is preferably accomplished with a drag and drop operation of one or more graphical objects from the input port window 82 to the output port window 82' of tree view screen 81. This operation has the effect of capturing, as indicated at 186, configuration data corresponding to the selected source of the data stream(s). For example, dragging and dropping the desired graphical objects enables element manager 22 to automatically capture corresponding configuration data for the desired routing events such as input port number and location, output port number and location, content stream PID to be routed and bit rate for the content streams to be routed. Additionally, information regarding the targeted output port (determined based on where the object is dropped) is also captured (188) by element manager 22 and includes, for example, the location of the targeted output port. This information enables element manager 22 to create default settings and to automatically perform PID aliasing at 186 so that there are no data stream conflicts as

the various streams are routed through transport multiplexer 20. The drag & drop editors 93 and 94 are then displayed by element manager 22 as indicated at 188. The operator can then select the particular component to be edited and, at 192, element manager 22 receives the selection and displays a component editor (95 for video streams or 95' for audio streams) with default information for possible editing. If the default data shown in the component editor 95 is acceptable to the operator, the "OK" button can be selected to cue the element manager to take further action. In particular, closing of the component editor window causes element manager 22 to gather the information from the GUI and to request the creation of various MIB table entries as shown at 194. The TMX executes the routing events in accordance with the updated MIB's at 196 and the GUI is appropriately updated by the element manager 22 as indicated at 198. From the operator's perspective, routing has been specified and performed simply by dragging and dropping an icon from the input tree to an output tree. In actuality, a variety of routing parameters have been specified with the assistance of the system as described in detail above.

[0055] If operator 10 wishes to modify the default and/or captured data, operator 10 has the ability to edit the information in detail for each of the components that comprise the content stream. In the example shown, operator 10 has selected program 1 (in general, an operator would select some type of graphical object, such as an icon or its associated text) shown in editor window 93 and a more detailed editor window 94 is displayed, the window showing the constituent components of the selected program. In the case of Figure 6, program 1 has been selected for editing and it includes one video component and one audio component.

[0056] Graphical user interface 80 preferably has the capability of identifying content streams using a variety of graphical objects which include icons, alphanumeric character strings, actual program names, etc. on the various screens. The content stream identification data is preferably carried within the media stream so that it can be consistently displayed throughout the graphical user interface regardless of which viewing screen is presented to operator 10. Restated, graphical user interface 80

preferably presents a consistent content stream name or symbol and can display it throughout the interface.

**[0057]** With continuing reference to Figure 6, selection of the "OK" button of window 94 closes the drag and drop window and opens the component editor windows corresponding to the selected components as indicated at 192. In this case, selection of a component to be edited further results in display of one of component editor windows 95 and 95' where operator 10 has the further ability to specify details such as bit rate, target PID, etc. for any of the components of the desired program. In this case, video editor window 95 and audio editor 95' are displayed for consideration and possible editing. This feature enables a user to more easily allocate bandwidth among the various content streams being routed so that maximum bandwidth utilization can be achieved.

**[0058]** Upon selection of the "OK" button of one of windows 95 or 95', the element manager 22 changes the MIB table data in accordance with the edited changes and instructs the TMX to execute the specified routing configuration. Configuration manager 46 then sequentially configures the targeted multiplexer and quantization level processor and enables the input processor, in that order, as indicated at 196.

**[0059]** The module activation order, when an output port is enabled, is an important aspect of the present invention. In order to effectively execute a routing event, the targeted multiplexer, quantization level processor and input processor should be activated in the order specified to minimize the possibility of the destabilizing the system. In particular, configuration manager 46 directs the targeted multiplexer to collect the designated PIDs and route them to the targeted output. Second, the configuration manager 46 must provide the quantization level processor 64 with the appropriate bit rate and PMT for the content stream to be routed. Third, configuration manager 46 should instruct the input processor to send all of the content streams with a particular PID to the multiplexer. This is preferably accomplished by performing PID aliasing and then sending the associated data to the multiplexer as a low voltage differential signal.

**[0060]** As noted above, module activation in an order other than that discussed above may lead to system instability. If, for example, the configuration

manager attempted to enable the input processor first, the multiplexer may begin to receive a content stream that it does not expect and this confusion may cause the multiplexer to crash. Similarly, removing a content stream (ceasing to route the stream to the port) should be performed in a predetermined order dictated by configuration manager 46. In particular, the sequence noted above should be reversed (deactivation of the input processor, deactivation of the QLP and, finally, deactivation of the multiplexer). If, for example, the multiplexer were disabled first, the multiplexer may still receive a content stream from the input processor and, once again, this condition may crash the multiplexer.

[0061] Turning now to Figure 7, this figure illustrates various system bandwidth utilization viewing capabilities in accordance with one preferred embodiment of the present invention. As shown therein, operator 10 initiates the view bandwidth utilization feature of the invention by selecting the bandwidth manager menu item from the upper portion of tree view screen 81. This enables element manager 22 to display the bandwidth manager screen at 208 and the TMX begins polling the system for bandwidth utilization data and waiting for queries for this data as shown at 210 – 212. As indicated more fully in the accompanying computer program provided at the end of this specification, the MIB tables enable monitoring of MPEG input/output bandwidth utilization information. In particular, the TMXinputPIDtable is used for input rate monitoring per PID. The TMXoutputPIDtable is used for output rate monitoring per PID.

[0062] In particular, message handler 45 begins polling input processor and output multiplexers for data that is used to update the MIB tables (capturing data from these two sources allows the bandwidth display to show a comparison between the input bandwidth and output bandwidth) and sends the data as SNMP data to element manager 22, as indicated at 214. Element manager 22 periodically queries the TMX for this information and at 216 displays this data on graphical user interface 80. It then returns to continue polling for new bandwidth utilization data at 214. In this way, bandwidth utilization data for all enabled ports is continually updated and can be displayed by graphical user interface 80 in real-time. Bandwidth data polling preferably ceases when operator 10 closes the bandwidth windows 96 and 96' such as by switching to the chassis



or tree view screens. At that point, the PID's for the enabled content streams are deleted from the MIB tables.

[0063] Upon receipt of bandwidth utilization data, graphical user interface 80 displays a bandwidth utilization screen 96. This screen preferably includes automatically rescaling x and y axes and an individual graphical object for each content stream being routed, each object preferably being a bandwidth bar (bars 97, 97' and 97" in the example shown). Each bandwidth bar shown in screen 96 preferably includes the following plural attributes: an output bandwidth utilization value 97a, an input bandwidth utilization value 97b, a maximum input bandwidth utilization value 97c and minimum input bandwidth utilization value 97d. In practice, changes in the bandwidth utilization are automatically displayed in bandwidth utilization screen 96 in real-time.

[0064] Bandwidth utilization screen 96 can include a number of user-friendly features to make the graphical user interface even more intuitive and useful. For example, operator 10 may be provided with the ability to select or deselect a legend display shown on the right hand portion of bandwidth utilization screen 96. Similarly, operator 10 preferably has the ability to select or deselect display of the minimum and maximum bandwidth utilization values. Furthermore, screen 96 preferably has the ability to display the same mnemonic identifiers for the various streams that are used in other screens such as the tree view screen. Restated, the graphical user interface preferably reflects a consistent identifier for each content stream throughout the system. Naturally, other identifiers could be used if desired. These identifiers are preferably sent with the content streams so that they can be detected and displayed in various screens. As noted above, the identifiers may be displayed as colored icons and/or alphanumeric character strings, etc.

[0065] After viewing bandwidth utilization screen 96, operator 10 may select one of the bandwidth bars to dynamically display more detailed information about the various components that make up the content stream for the selected bar. For example, a given program might include one video and two audio components. Selecting a bandwidth bar will cause detailed bandwidth utilization window 96' (with additional

information about these components) to appear on the screen. This type of selection causes element manager 22 to generate a query at 216 which is responded to by the TMX at 210/212. As shown in window 96', the program name, the group ID and the total bandwidth at the instant that the bandwidth bar was taken are captured and displayed on the screen. In this illustrative embodiment, the bandwidth bar for program 2 was selected when the bandwidth utilization was about five megabits per second (compare windows 96 and 96' of Figure 7). Additionally, the detailed window breaks the selected program down into its constituent components. In this case the program has three constituent parts: IP data 1, video data 1 and audio data 1.

[0066] The screen 96' shows even more detailed information for each component of the program. This information preferably includes a bandwidth minima value, a bandwidth maxima value and the instantaneous bandwidth utilization of the constituent components at the instant the detailed bandwidth utilization window was selected. With joint reference to screens 96 and 96', it will be appreciated that the displayed bandwidth utilization of the constituent components sums to the bandwidth utilization of the entire program. Additionally, the sum of the minimum values of the constituent components equals the minimum value for the program as a whole. Similarly, the maximum value for the entire program equals the sum of the minimum values for each of the constituent components. Finally, the display shows the packet identifier PID associated with the program.

[0067] Since this aspect of the system displays bandwidth in real-time, the operator will see the bandwidth utilization varying over time. Also, differences in bandwidth utilization at different points in time will reflect the fact that input signals can vary over time on the input side of the whole system. For example, if an input signal suddenly includes an additional component, the bandwidth display screen will reflect that change in real-time.

[0068] Figure 8 illustrates various event logging and viewing capabilities 222 in accordance with one preferred embodiment of the present invention. As shown therein, the system has the ability to filter the log messages displayed on the graphical

user interface. Viewing log information in accordance with the present invention initially entails operator selection of an appropriate log filter level, thereby placing the system into one of four modes. The filter level is recorded by the element manager 22 and the number and type of messages displayed in the log message window 87 of graphical user interface 80 is dictated by the filter level. The desired log filter level can be selected from the "view" drop down menu item near the top of tree view screen 81 and then selecting the log messages option. There are preferably four filter levels: normal status, emergency status, fault status and debug. In debug mode all of the generated log messages are displayed.

[0069] Upon startup, the TMX chassis 42 the status query task begins to poll the system to thereby generate log messages that are used to populate the TMXLogPortTable, as shown at 224. The SNMP agent 44 then waits to respond to for queries for this information as shown at 226. This log messages can be generated by any one of the various firmware modules and element manager 22, GUI 80 and TMX chassis 42 cooperate to continually pass log messages in accordance with the previously selected log level to the graphical user interface for display in the scrolling log message window. Additionally, these log messages are stored for possible retrieval and analysis in the future. Although the log messages presented to an operator in normal use can be filtered, all log messages generated by the system are preferably stored on the element manager's host computer. One separate log file is preferably generated for each day the system is in use and operator 10 has the ability to retrieve and view log messages for any given day in the log file archive screen 98.

[0070] Upon selection of the Log File Menu by operator 10, element manager 22 retrieves, displays and stores log files as indicated at 228. This screen is accessed by selecting the "view" menu item near the top of the tree view screen 81 and by then selecting the appropriate option. Upon selection of one of the daily log files from the list of log files in the archive screen 98, individual log messages from the selected log file are displayed for viewing on screen 98' as indicated at 230. When reviewing stored

log messages, the operator also has the ability to filter the information by selecting one of the four filter levels as discussed above.

[0071] Figure 9 illustrates various future content stream routing capabilities 238 in accordance with one preferred embodiment of the present invention. Specification of future event(s) is initially driven by operator action on the tree view screen. In particular, upon initialization and discovery of the system, the system initially sets up one routing event that spans the current time up to a predetermined time in the future (e.g., two years). This is shown in a time bar 99. Operator 10 can then select time bar 99, as shown in the upper right hand portion of tree view screen 81. The resulting pop-up menu allows operator 10 to either modify the displayed current event or to create a new event. In the case of specifying the future routing events, operator 10 would create a new event by selecting the create new event option and by specifying start and stop times for the new event. At that point, indicated at 240, another duplicate event (by default) is created by element manager 22. This information is then sent to the graphical user interface 80 for display and possible modification as shown at 241. The particular editor that is presented to operator 10 depends on what type of event will be created. In the representative example of Figure 9, audio and video editors 95 and 95' are presented. IP data streams could also be specified for a future routing event as will readily occur to those of ordinary skill based on the teachings contained herein. Once all of the various details for the various components of the future event has been completed, this information will be gathered by the element manager at 242 and displayed on screen 81'. As shown on screen 81', three events have been define in the illustrative example of Figure 9. At 244, element manager 22 requests that new entries be added to certain MIB's and TMX chassis 42 executes the configuration changes at 246. Also, element manager 22 updates the GUI at 252. This results in a tree view screen 81" that is substantially similar to that of screen 81', but that displays the routing trees according to the newly executed configuration.

[0072] Preferably, none of this future event configuration data is provided to TMX chassis 42 until shortly prior to the predetermined time for commencement of the

newly defined future event. Then (e.g., about 30 seconds prior to the predetermined time), the entire configuration data is sent to TMX chassis 42 for execution. This routing event data is slightly different from that discussed above with respect to Figure 6, in that it also includes predetermined time data indicating when the new routing configuration is to occur. In this way operator 10 can configure the system to automatically change configuration routing control at future predetermined points in time, even in the absence of the operator. Thus, the system permits automated control of the inventive broadband media hardware by predetermining routing configuration information for extended time periods and enabling automatic execution of such configuration changes.

[0073] Figure 10 illustrates various IP data encapsulation and insertion capabilities and processes 260 in accordance with one preferred embodiment of the present invention. As described in greater detail below, the present invention enables operator 10 to define and execute IP data encapsulation either manually or semi-automatically.

[0074] In particular, the preferred embodiment of the present invention provides operators with the ability to manually enter IP encapsulation configuration data element by element or, alternatively, to automatically enter IP encapsulation configuration data by dragging and dropping graphical objects to and from various locations of the tree view screen 81. Element manager 22 cooperates with graphical user interface 80 to execute the various routing commands specified by corresponding drag and drop operations. This is achieved with automated population of MIB tables corresponding to the various fields necessary to define a routing command. Drag and drop operations on graphical user interface 80 assist operator 10 in defining IP encapsulation specifications for the system in a manner substantially analogous to the semi-automatic definition of video and audio routing events shown and described with reference to Figure 6. Those of ordinary skill in the art will readily appreciate how to extend these concepts to implement drag and drop procedures in order to achieve semi-automated IP data encapsulation based on the teachings of this specification. Manual, or element by

element, IP data encapsulation techniques are described immediately below with respect to Figures 10 and 11.

[0075] With primary reference to Figure 10, operator 10 can specify one or more IP data encapsulation events 260 by selecting the graphical objects representing a desired location (e.g., an enabled output port) from a tree view screen 262. Operator 10 can then select a particular program into which encapsulated IP data will be inserted. This enables element manager 22 to capture configuration data relating to the targeted output port and any programs that may be resident thereon at 264. In the representative example of Figure 10, program 1 has been selected for insertion of an IP data component. Responsive to operator selection of program 1, element manager 22 (at 266) displays a program editor 270 and sends default output port values from the to the graphical user interface for display. Operator 10 can then enter various values relating to a program into which an IP data component will be inserted with the assistance of element manager 22 at 272. General and detailed IP data component editors 274 will then be displayed so that a variety of other parameters can be specified by operator 10. Operator 10 has the ability to edit the add/remove/change detailed information in the IP data components editors for each of the components that comprise the content stream. In particular, operator 10 has the ability to specify details such as source and destination IP addresses, bit rate, target PID, etc. for each component of the selected program in the general and detailed editor windows 274. This feature enables a user to more easily allocate bandwidth among the various IP data streams being created so that maximum bandwidth utilization can be achieved. Up to 128 IP data streams may be simultaneously specified for encapsulation and insertion in this way.

[0076] Upon selection of the "OK" button of one of windows 274, element manager 22 executes a number of functions at 276. In particular, element manager 22 gathers the edited information from the GUI and requests that various new entries be placed into certain MIB tables with default and/or edited data (as shown at 276). Element manager 22 also provides this information to TMX 42 for execution as shown at 278 of Figure 10 and in Figure 11. In particular, at 278, SNMP agent 44 creates the new MIB

entries, message handler 45 passes the information to configuration manager 46 which configures one or more multiplexers and instructs the IP encapsulation module 66 to begin collecting IP data. IP encapsulation module 66 then receives IP data from the specified source IP address, encapsulates each IP data packet as one or more MPEG packets, to thereby form MPEG data streams, and sends them to the targeted multiplexer(s). The targeted multiplexer(s) receives the assembled MPEG data packets and stream the MPEG data appropriately. At 280, the element manager updates the graphical user interface 80 which displays the updated information on tree view screen 289. Operator 10 can then view an IP data icon 290 that indicates encapsulation and insertion of IP data is occurring.

[0077] The portion of block 278 that performs the IP encapsulation process is illustrated in detail in Figure 11. As shown therein, upon execution of IP encapsulation process 282, the encapsulation module 66 instructs IP data stack (of the operating system running on the host processor) to collect/receive and examine an IP data packet at 292. At 293 the module 66 then verifies that the system is prepared to process IP packets (e.g., the target multiplexer(s) has/have been properly configured). The destination IP address for the received IP data packet is then tested for validity at 294. In particular, the destination IP address is checked to determine whether it is the broadcast, unicast or multicast IP address. This is preferably accomplished by verifying that the destination address is within the multicast range and that the address has been specified for data collection/reception. If the IP address indicates that the IP packet is not a multicast packet, the determination is made that the IP data packet must be either a broadcast or unicast packet. If so, the data packet is passed through the operating system (OS) stack in a conventional manner and the process passes to 296 where it simply waits to receive the next IP data packet. In particular, the preferred OS (VxWorks) employs a standard seven-layer OSI compliant IP stack that processes each broadcast and/or unicast packet to determine its type and the application that it should process it. Thus, for example, a broadcast packet that is found to be an ARP request would be sent to the ARP task for processing.

**[0078]** Conversely, if the source IP address indicates that the IP data packet is a multicast IP packet, the packet cannot simply be routed through the OS stack because the OS will not recognize the data packet except in the unlikely event that it is the intended recipient of the packet. Thus, if the IP address indicates that the data packet is a multicast packet and if that address is one of the 128 addresses that element manage 22 has indicated as being associated with IP data that is to be encapsulated, the IP data will be converted to a different form and routed without going through the IP stack as an IP data packet. To achieve this, the process first passes to 297 where the IP data packet is fragmented, if necessary, into smaller content components for processing. The process then passes to 298 where an MPEG data packet is assembled and sent to the appropriate multiplexer(s). In particular, a 4 byte MPEG header that includes the target PID for this packet is created at 300. Then, at 302 the destination IP address is extracted from the IP data packet and used to create a 16 byte DSM-CC (Data Storage Media Command and Control) header for the first MPEG data packet. A conventional 4 byte Cyclic Redundancy Code (CRC or CRC32) MPEG suffix is preferably also included in the last MPEG packet (e.g., following the last byte of content). Since the system can support output data in either one of DVB or ASTC data formats, the DSM-CC header also indicates which format the output data is in to thereby account for the differences between these formats. At 304, up to 168 bytes of content are added to the MPEG 188 byte packet being created. If this can hold all of the content to be sent, then a CRC is appended after the last byte of content. At 308, a determination is made whether any fill data is needed to complete the MPEG packet. If so, process 282 passes to 310 where the remainder of the MPEG packet is filled with dummy data. This data is preferably the numerical value of 255 (FF in hexadecimal) and is repeated until a complete 188 byte MPEG data packet has been formed. With this system of the preferred embodiment, a maximum of one IP data packet will be inserted into a single MPEG packet. If no fill is needed (or after the packet has been filled), the process passes to 312 where the assembled packet is sent to the targeted multiplexer and it is preferably stored in a FIFO for combination with additional MPEG packets, if any. Also, the process passes to 314



where it is determined whether or not the received IP data packet has been fully encapsulated. If so, the process passes to 316 where the multiplexer receives an indication of how many MPEG data packets it has received together with an indication that this/these packet(s) should be transmitted. The process 282 then passes to 296 where the IP encapsulation module waits for the next IP data packet to be encapsulated.

**[0079]** If it is determined at 314 that the IP data packet has not been fully encapsulated, process 282 passes to 318 where additional content from the IP data packets are assembled into MPEG data packets and sent to the appropriate multiplexer. In particular, process 282 passes from 314 to 320 where an MPEG header for the next MPEG data packet is created. Up to 184 bytes of IP data and CRC (for the last MPEG packet) are then added to the packet at 322 and, at 326, a determination is made whether any fill data is needed to complete the MPEG packet. If so, process 282 passes to 328 where the remainder of the MPEG packet is filled with dummy data. This data is also preferably the numerical value of 255 (FF in hexadecimal) and is repeated until a complete 188 byte MPEG data packet has been formed. If no fill is needed (or after the packet has been filled), process 282 passes to 330 where the assembled packet is sent to the targeted multiplexer and it is preferably stored in a FIFO for combination with all prior and subsequent assembled MPEG packets, if any. Also, the process passes to 332 where it is determined whether or not the received IP data packet has been fully encapsulated. If not, steps 320 through 332 are repeated until the entire IP data packet has been encapsulated and, eventually, the process passes to 316 and 296 as noted immediately below. If so, the multiplexer receives an indication of how many MPEG data packets it has received together with an indication that these packets should be transmitted at 316. The process then passes to 296 where the IP encapsulation module waits for the next multicast IP data packet to be encapsulated. Process 282 terminates when operator 10 specifies a different function for the subject output port or when the time period for the specified event has expired. At that point, IP encapsulation module 66 awaits further instructions from configuration manager 46.

[0080] The following computer program listing sets forth the TMX-MIB definitions referred to above:

```

TMX-MIB DEFINITIONS ::= BEGIN
IMPORTS
    NetworkAddress, IpAddress, Gauge, TimeTicks
        FROM RFC1155-SMI
    RowStatus, DisplayString, DateAndTime, TEXTUAL-CONVENTION
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
    ifIndex
        FROM RFC1213-MIB
    Counter32, Integer32, OBJECT-TYPE, MODULE-IDENTITY FROM SNMPv2-SMI
    bmr FROM NLS-BBNIDENT-MIB;

tmxMIB MODULE-IDENTITY
    LAST-UPDATED "0112202341Z"
    ORGANIZATION "Motorola BCS"
    CONTACT-INFO "John Sanders, Ext 3505 4A31"
    DESCRIPTION
        "TMX MIB Version 1.1a"
    REVISION "0008141355Z"
    DESCRIPTION
        ""
    ::= { bmr 1 }

ActionTriggerType ::= TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
        "Type of trigger mechanism for execution of
        an operation"
    SYNTAX BITS {
        operationCreation(0),
        activationTime(1),
        spliceImmediateFlag(2)
    }

org OBJECT IDENTIFIER
    ::= { iso 3 }

dod OBJECT IDENTIFIER
    ::= { org 6 }

internet OBJECT IDENTIFIER
    ::= { dod 1 }

private OBJECT IDENTIFIER
    ::= { internet 4 }

enterprises OBJECT IDENTIFIER
    ::= { private 1 }

tmxMIBObjects OBJECT IDENTIFIER
    ::= { tmxMIB 1 }

tmxMIBConformance OBJECT IDENTIFIER
    ::= { tmxMIB 2 }

```

```

tmxMediaInterfaces OBJECT IDENTIFIER
    ::= { tmxMIBObjects 1 }

tmxMediaControl OBJECT IDENTIFIER
    ::= { tmxMIBObjects 2 }

tmxMediaMonitor OBJECT IDENTIFIER
    ::= { tmxMIBObjects 3 }

-- TMX specific extensions to the MIB 2 Interfaces Table

tmxNetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF TmxNetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        " This is the network table associating TMX interfaces (ifIndex)
          with other types of equipment."
    ::= { tmxMediaInterfaces 1 }

tmxNetEntry OBJECT-TYPE
    SYNTAX      TmxNetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "tmxNetTable entry"
    INDEX       { ifIndex, tmxNetIndex }
    ::= { tmxNetTable 1 }

TmxNetEntry ::= SEQUENCE {
    tmxNetIndex          Integer32,
    tmxNetEqpType        BITS,
    tmxNetEqpName        DisplayString,
    tmxNetEqpIpAddress   IpAddress,
    tmxNetEqpOperationalState Integer32,
    tmxNetEqpAlarmStatus Integer32,
    tmxNetEqpHeartBeatMonitor IpAddress,
    tmxNetEqpRedundancyGroup DisplayString,
    tmxNetEqpBackUp      IpAddress,
    tmxNetEntryStatus    RowStatus
}

tmxNetIndex OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Identifier index for this networked equipment."
    ::= { tmxNetEntry 1 }

tmxNetEqpType OBJECT-TYPE
    SYNTAX      BITS { se1000(0), se2000(1), tmx(2),
                      dataServer(3), adServer(4), computer(5) }
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Type of equipment adjacent on ifIndex interface."

```

```
::= { tmxNetEntry 2 }
```

```
tmxNetEqpName      OBJECT-TYPE
    SYNTAX          DisplayString
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Unique user assigned name to the physical equipment."
 ::= { tmxNetEntry 3 }
```

```
tmxNetEqpIpAddress OBJECT-TYPE
    SYNTAX          IPAddress
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "IP Address of the adjacent equipment."
 ::= { tmxNetEntry 4 }
```

```
tmxNetEqpOperationalState OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "OperationalState of this networked equipment."
 ::= { tmxNetEntry 5 }
```

```
tmxNetEqpAlarmStatus      OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "AlarmStatus of this networked equipment."
 ::= { tmxNetEntry 6 }
```

```
tmxNetEqpHeartBeatMonitor OBJECT-TYPE
    SYNTAX          IPAddress
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Heart beat monitor for this networked equipment."
 ::= { tmxNetEntry 7 }
```

```
tmxNetEqpRedundancyGroup OBJECT-TYPE
    SYNTAX          DisplayString
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Unique name of the redundancy group of this networked
equipment."
 ::= { tmxNetEntry 8 }
```

```
tmxNetEqpBackUp      OBJECT-TYPE
    SYNTAX          IPAddress
```

```

MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "IP Address of the back up of this networked equipment."
::= { tmxNetEntry 9 }

tmxNetEntryStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION "Status of this entry."
    ::= { tmxNetEntry 10 }

-- tmxMediaControl consists of tables used to configure the TMX media control
-- functionality.

tmxOpNum OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION "The number of last Op created."
    ::= { tmxMediaControl 1 }

tmxToaLock OBJECT-TYPE
    SYNTAX      BITS { unlock(0), lock(1)}
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION "The number of last Op created."
    ::= { tmxMediaControl 2 }

tmxToaLockTime OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION "The time (in second) to wait before starts executing the
unlock operation"
    ::= { tmxMediaControl 3 }

tmxDeleteAll OBJECT-TYPE
    SYNTAX      BITS { delete(0), keep(1)}
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION "The flag indicates the deletion of the entire media control
database"
    ::= { tmxMediaControl 4 }

tmxReserved OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION "The number of last Op created."
    ::= { tmxMediaControl 5 }
*****
tmxIfTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF TmxIfEntry
    MAX-ACCESS  not-accessible

```

```

STATUS          current
DESCRIPTION
    " This is the TMX specific extension of ifTable."
::= { tmxMediaControl 6 }

tmxIfEntry      OBJECT-TYPE
    SYNTAX       TmxIfEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION   "tmxIfTable entry"
    INDEX        { ifIndex }
    ::= { tmxIfTable 1 }

TmxIfEntry ::= SEQUENCE {
    tmxIfName          DisplayString,
    tmxIfTSId          Integer32,
    tmxIfType          BITS,
    tmxIfASIMode       BITS,
    tmxIfStandard      BITS,
    tmxIfTransportBitRate Integer32,
    tmxIfActionTrigger ActionTriggerType,
    tmxIfActionTime    DateAndTime,
    tmxIfIQ            BITS,
    tmxIfSttDestinationOffset Integer32,
    tmxIfSttDsStatus   BITS,
    tmxIfSttDsDay      Integer32,
    tmxIfSttDsHour     Integer32,
    tmxIfPacketLength  BITS,
    tmxIfEntryStatus   RowStatus
}

*****

tmxIfName      OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "User assigned name to the physical interface."
    ::= { tmxIfEntry 1 }

tmxIfTSId      OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "Mpeg Transport Identifier associated with this physical
interface, if any."
    ::= { tmxIfEntry 2 }

tmxIfType      OBJECT-TYPE
    SYNTAX      BITS { asi(0), ds3-ansi(1), ds3-fsi(2),
                      dhei(3), dhei-high(4),
                      smpte310(5), ds3-rmi(6) }
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "tmxIfType."
    ::= { tmxIfEntry 3 }

```

```

tmxIfASIMode          OBJECT-TYPE
    SYNTAX              BITS { burst(0), byte(1)}
    MAX-ACCESS           read-create
    STATUS               current
    DESCRIPTION          "Output ASI mode, burst or byte."
    ::= { tmxIfEntry 4 }

tmxIfStandard          OBJECT-TYPE
    SYNTAX              BITS { atsc(0), dvb(1), mpeg2(2), none(3), dcii(4)}
    MAX-ACCESS           read-create
    STATUS               current
    DESCRIPTION          "Output ASI standard."
    ::= { tmxIfEntry 5 }

tmxIfTransportBitRate  OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS           read-create
    STATUS               current
    DESCRIPTION          "Transport bit rate in Mbps."
    ::= { tmxIfEntry 6 }

tmxIfActionTrigger     OBJECT-TYPE
    SYNTAX              ActionTriggerType
    MAX-ACCESS           read-create
    STATUS               current
    DESCRIPTION          "Trigger type that will initiate this operation."
    ::= { tmxIfEntry 7 }

tmxIfActionTime        OBJECT-TYPE
    SYNTAX              DateAndTime
    MAX-ACCESS           read-create
    STATUS               current
    DESCRIPTION          "If trigger type is a time of action this object contains
the time."
    ::= { tmxIfEntry 8 }

tmxIfIQ                OBJECT-TYPE
    SYNTAX              BITS {combined(0), i-out(1), q-out(2), iq-split(3)}
    MAX-ACCESS           read-create
    STATUS               current
    DESCRIPTION          "Select IQ mux mode for this transport stream, combined, I,
Q or I&Q."
    ::= { tmxIfEntry 9 }

tmxIfSttDestinationOffset OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS           read-create
    STATUS               current

```



## DESCRIPTION

"Offset from UTC as used in the STT. This may not be the same value as used by the TMX system and may vary between transport streams depending on destination"

```
::= { tmxIfEntry 10 }
```

```
tmxIfSttDsStatus      OBJECT-TYPE
```

```
SYNTAX                BITS { not-in-ds(0), in-ds(1) }
```

```
MAX-ACCESS             read-create
```

```
STATUS                 current
```

```
DESCRIPTION            "Daylight savings status as used in the STT."
```

```
::= { tmxIfEntry 11 }
```

```
tmxIfSttDsDay         OBJECT-TYPE
```

```
SYNTAX                Integer32
```

```
MAX-ACCESS             read-create
```

```
STATUS                 current
```

```
DESCRIPTION
```

"Local day of month daylight savings status will change (1-31) as used in the STT."

```
::= { tmxIfEntry 12 }
```

```
tmxIfSttDsHour        OBJECT-TYPE
```

```
SYNTAX                Integer32
```

```
MAX-ACCESS             read-create
```

```
STATUS                 current
```

```
DESCRIPTION
```

"Local hour of day daylight savings status will change (0-18) as used in the STT."

```
::= { tmxIfEntry 13 }
```

```
tmxIfPacketLength     OBJECT-TYPE
```

```
SYNTAX                BITS { pkt188(0), pkt204(1) }
```

```
MAX-ACCESS             read-create
```

```
STATUS                 current
```

```
DESCRIPTION            "Transport stream packet length."
```

```
::= { tmxIfEntry 14 }
```

```
tmxIfEntryStatus      OBJECT-TYPE
```

```
SYNTAX                RowStatus
```

```
MAX-ACCESS             read-create
```

```
STATUS                 current
```

```
DESCRIPTION            "Status of this entry."
```

```
::= { tmxIfEntry 15 }
```

\*\*\*\*\*

```
tmxIfGiExtTable       OBJECT-TYPE
```

```
SYNTAX                SEQUENCE OF TmxIfGiExtEntry
```

```
MAX-ACCESS             not-accessible
```

```
STATUS                 current
```

```
DESCRIPTION
```

" This is the extension table defining the DS3-GI TMX interface parameters."

```
::= { tmxMediaControl 7 }
```

```

tmxIfGiExtEntry      OBJECT-TYPE
    SYNTAX             TmxIfGiExtEntry
    MAX-ACCESS         not-accessible
    STATUS             current
    DESCRIPTION
        "tmxIfGiExtTable entry"
    INDEX              { ifIndex }
    ::= { tmxIfGiExtTable 1 }

```

```

TmxIfGiExtEntry ::= SEQUENCE {
    tmxIfGiExtCodeRate      BITS,
    tmxIfGiExtSymbolRate    BITS,
    tmxIfGiExtModType        BITS,
    tmxIfGiExtPowerLevel    Integer32,
    tmxIfGiExtCarrierFrq    Integer32,
    tmxIfGiExtCwMode         BITS,
    tmxIfGiExtMute           BITS,
    tmxIfGiExtEntryStatus    RowStatus
}

```

\*\*\*\*\*

```

tmxIfGiExtCodeRate    OBJECT-TYPE
    SYNTAX             BITS { cr-5-11(0), cr-1-2(1), cr-3-5(2),
                             cr-2-3(3), cr-3-4(4), cr-4-5(5),
                             cr-5-6(6), cr-7-8(7) }
    MAX-ACCESS         read-create
    STATUS             current
    DESCRIPTION
        "Code rate: 5/11, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6,
        7/8"
    ::= { tmxIfGiExtEntry 1 }

```

```

tmxIfGiExtSymbolRate  OBJECT-TYPE
    SYNTAX             BITS { sr-38-8(0), sr-29-27(1),
                             sr-19-51(2), sr-14-63(3),
                             sr-11-71(4), sr-9-76(5),
                             sr-7-32(6), sr-4-88(7),
                             sr-3-25(8), sr-2-44(9),
                             sr-1-83(10) }
    MAX-ACCESS         read-create
    STATUS             current
    DESCRIPTION
        "Symbol rate: 38.8 29.27, 19.51, 14.63, 11.71,
        9.76, 7.32, 4.88, 3.25, 2.44, 1.83"
    ::= { tmxIfGiExtEntry 2 }

```

```

tmxIfGiExtModType     OBJECT-TYPE
    SYNTAX             BITS { qpsk(0), bpsk(1), oqpsk(2) }
    MAX-ACCESS         read-create
    STATUS             current
    DESCRIPTION
        "Modulation type."
    ::= { tmxIfGiExtEntry 3 }

```

```

tmxIfGiExtPowerLevel  OBJECT-TYPE

```

```

SYNTAX          Integer32
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "Power level: 0-20 representing -5dBm to -15dBm in 0.5dBm
steps."
::= { tmxIfGiExtEntry 4 }

tmxIfGiExtCarrierFrq      OBJECT-TYPE
SYNTAX          Integer32
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "47000 kHz - 93000 kHz for the 70 MHz band and
104000 khz - 176000 kHz for the 140 MHz band.
The frequency must be in steps of 125 kHz."
::= { tmxIfGiExtEntry 5 }

tmxIfGiExtCwMode          OBJECT-TYPE
SYNTAX          BITS { modulated(0), unmodulated(1) }
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "IF carrier to be unmodulated."
::= { tmxIfGiExtEntry 6 }

tmxIfGiExtMute            OBJECT-TYPE
SYNTAX          BITS { notMuted(0), muted(1) }
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "IF carrier to be muted."
::= { tmxIfGiExtEntry 7 }

tmxIfGiExtEntryStatus     OBJECT-TYPE
SYNTAX          RowStatus
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "Status of this entry."
::= { tmxIfGiExtEntry 8 }

*****
tmxStatGroupTable          OBJECT-TYPE
SYNTAX          SEQUENCE OF TmxStatGroupEntry
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION     "The StatGroup Control table. Each row in this
table represents a create/delete Op command. All
Ops with the same Time of Action are to be executed
in the order specified by the message index.
Program elements within a StatGroup can be
Constant Bit Rate (CBR) in which case their minBw=maxBw,
Variable Bit Rate (VBR) in which case their minBw<maxBw,
and opportunistic in which case they use up only spare Bw
left unused after all CBR and VBR programs are multiplexed"
::= { tmxMediaControl 8 }

```

```

tmxStatGroupEntry    OBJECT-TYPE
    SYNTAX             TmxStatGroupEntry
    MAX-ACCESS         not-accessible
    STATUS             current
    DESCRIPTION
        "StatGroupTable entry"
    INDEX              { ifIndex, tmxStatGroupOpIndex }
    ::= { tmxStatGroupTable 1 }

```

```

TmxStatGroupEntry ::= SEQUENCE {
    tmxStatGroupOpIndex      Integer32,
    tmxStatGroupMsg          BITS,
    tmxStatGroupId           Integer32,
    tmxStatGroupSize         Integer32,
    tmxStatGroupTranscBwPercent Integer32,
    tmxStatGroupMaxInputBw   Integer32,
    tmxStatGroupMaxOutputBw Integer32,
    tmxStatGroupActionTrigger ActionTriggerType,
    tmxStatGroupActionTime   DateAndTime,
    tmxStatGroupStatus       RowStatus
}

```

\*\*\*\*\*

```

tmxStatGroupOpIndex    OBJECT-TYPE
    SYNTAX             Integer32
    MAX-ACCESS         not-accessible
    STATUS             current
    DESCRIPTION
        "This is used for unique identification and sequencing of all
        operational configuration messages. The value is the value of
        tmxOpNum when entry was created and is set by the manager."
    ::= { tmxStatGroupEntry 1 }

```

```

tmxStatGroupMsg        OBJECT-TYPE
    SYNTAX             BITS { create ( 0 ) , delete ( 1 ) }
    MAX-ACCESS         read-create
    STATUS             current
    DESCRIPTION
        "Each table row constitutes a command message. The message
        type indicates whether the Command type is to either create
        or delete the item described in this row. Any create command
        is further restricted by the Action Trigger Type. Depending on
        the Action Trigger Type, the command will either be immediately
        executed, executed at a time specified by Time of Action, or
        executed when a SPliceImmediateFlag is raised."
    ::= { tmxStatGroupEntry 2 }

```

```

tmxStatGroupId         OBJECT-TYPE
    SYNTAX             Integer32
    MAX-ACCESS         read-create
    STATUS             current
    DESCRIPTION
        "Group identifier in control multiplex."
    ::= { tmxStatGroupEntry 3 }

```

```

tmxStatGroupSize      OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION         "Number of members in group."
    ::= { tmxStatGroupEntry 4 }

```

```

tmxStatGroupTranscBwPercent  OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION         "Percentage of overall bandwidth to allocate to multiplex group,
                        if transcoded."
    ::= { tmxStatGroupEntry 5 }

```

```

tmxStatGroupMaxInputBw      OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION         "Define maximum bandwidth to allocate to this input stat group,
                        if stat group is input."
    ::= { tmxStatGroupEntry 6 }

```

```

tmxStatGroupMaxOutputBw     OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION         "Define maximum bandwidth to allocate to this output stat group,
                        if stat group is input."
    ::= { tmxStatGroupEntry 7 }

```

```

tmxStatGroupActionTrigger  OBJECT-TYPE
    SYNTAX              ActionTriggerType
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION         "Trigger type that will initiate this operation."
    ::= { tmxStatGroupEntry 8 }

```

```

tmxStatGroupActionTime      OBJECT-TYPE
    SYNTAX              DateAndTime
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION         "If trigger type is a time of action this object contains the
time."
    ::= { tmxStatGroupEntry 9 }

```

```

tmxStatGroupStatus          OBJECT-TYPE
    SYNTAX              RowStatus

```

```

MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "Status of this row."
::= { tmxStatGroupEntry 10 }

```

\*\*\*\*\*

```

tmxProgTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF TmxProgEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The MPEG Service/Program Control table. Each row in this
        table represents a create/delete Op command. All
        Ops with the same Time of Action are to be executed
        in the order specified by the message index.
        This table is used to create programs in the output
        multiplex and to associate programs with possibly
        external input sources. It is not used to route a
        complete program from input to output. All routing is done
        by the Program Component Table."
    ::= { tmxMediaControl 9 }

```

```

tmxProgEntry OBJECT-TYPE
    SYNTAX      TmxProgEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Entry into the Program Control table."
    INDEX       { ifIndex, tmxStatGroupOpIndex, tmxProgOpIndex }
    ::= { tmxProgTable 1 }

```

```

TmxProgEntry ::= SEQUENCE {
    tmxProgOpIndex      Integer32,
    tmxProgMsg          BITS,
    tmxProgGrpId        Integer32,
    tmxProgEncIPAddr    IPAddress,
    tmxProgEncPort      Integer32,
    tmxProgTmxIPAddr    IPAddress,
    tmxProgTmxPort      Integer32,
    tmxProgProgId       Integer32,
    tmxProgProgName     DisplayString,
    tmxProgPcrPID       Integer32,
    tmxProgGIDS3IQ      BITS,
    tmxProgActionTrigger ActionTriggerType,
    tmxProgActionTime   DateAndTime,
    tmxProgStatus       RowStatus
}

```

\*\*\*\*\*

```

tmxProgOpIndex OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This is used for unique identification and sequencing of all
        operational configuration messages. The value is the value of
        tmxOpNum when entry was created and is set by the manager."
    ::= { tmxProgEntry 1 }

```

```

tmxProgMsg          OBJECT-TYPE
    SYNTAX            BITS { create ( 0 ) , delete ( 1 ) }
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "Each table row constitutes a command message. The message
        type indicates whether the Command type is to either create
        or delete the item described in this row. Any create command
        is further restricted by the Action Trigger Type. Depending on
        the Action Trigger Type, the command will either be immediately
        executed, executed at a time specified by Time of Action, or
        executed when a SPliceImmediateFlag is raised."
    ::= { tmxProgEntry 2 }

tmxProgGrpId        OBJECT-TYPE
    SYNTAX            Integer32
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "Statistical Group identifier in output multiplex,
        if part of a statistical group."
    ::= { tmxProgEntry 3 }

tmxProgEncIPAddr    OBJECT-TYPE
    SYNTAX            IpAddress
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "If any, IP address of the encoder for this service for
        distributed stat mux."
    ::= { tmxProgEntry 4 }

tmxProgEncPort      OBJECT-TYPE
    SYNTAX            Integer32
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "If any, Port number of the encoder for this service for
        distributed stat mux."
    ::= { tmxProgEntry 5 }

tmxProgTmxIPAddr    OBJECT-TYPE
    SYNTAX            IpAddress
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "If any, IP address of the TMX for this service for distributed
        stat mux."
    ::= { tmxProgEntry 6 }

tmxProgTmxPort      OBJECT-TYPE
    SYNTAX            Integer32
    MAX-ACCESS        read-create

```

```

STATUS          current
DESCRIPTION
    "If any, Port number of the TMX for this service for distributed
    stat mux."
::= { tmxProgEntry 7 }

```

```

tmxProgProgId    OBJECT-TYPE
SYNTAX          Integer32
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Output MPEG program (i.e. service) number."
::= { tmxProgEntry 8 }

```

```

tmxProgProgName   OBJECT-TYPE
SYNTAX          DisplayString
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Output program name assigned by the user."
::= { tmxProgEntry 9 }

```

```

tmxProgPcrPID     OBJECT-TYPE
SYNTAX          Integer32
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "PCR PID associated with the program."
::= { tmxProgEntry 10 }

```

```

tmxProgGIDS3IQ    OBJECT-TYPE
SYNTAX          BITS { splitI ( 0 ) , splitQ ( 1 ) , combined (2) }
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Each table row constitutes a command message."
::= { tmxProgEntry 11 }

```

```

tmxProgActionTrigger OBJECT-TYPE
SYNTAX          ActionTriggerType
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Trigger type that will initiate this operation."
::= { tmxProgEntry 12 }

```

```

tmxProgActionTime OBJECT-TYPE
SYNTAX          DateAndTime
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "If trigger type is a time of action this object contains the
time."

```



```

::= { tmxProgEntry 13 }

tmxProgStatus      OBJECT-TYPE
    SYNTAX          RowStatus
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Status of the row."
::= { tmxProgEntry 14 }
*****
tmxProgCompTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF TmxProgCompEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The MPEG Service/Program Component Control table. Each row in this
        table represents a create/delete Op command. All
        Ops with the same Time of Action are to be executed
        in the order specified by the message index."
    ::= { tmxMediaControl 10 }

tmxProgCompEntry   OBJECT-TYPE
    SYNTAX          TmxProgCompEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION     "Entry"
    INDEX           { ifIndex, tmxStatGroupOpIndex, tmxProgOpIndex ,
tmxProgCompOpIndex }
    ::= { tmxProgCompTable 1 }

TmxProgCompEntry ::= SEQUENCE {
    tmxProgCompOpIndex      Integer32,
    tmxProgCompMsg          BITS,
    tmxProgCompTpeId        Integer32,
    tmxProgCompSrcId        Integer32,
    tmxProgCompPID          Integer32,
    tmxProgCompAliasPID     Integer32,
    tmxProgCompStreamType   BITS,
    tmxProgCompPriority     Integer32,
    tmxProgCompTrcMode      BITS,
    tmxProgCompEmbeddedPCR  BITS,
    tmxProgCompActionTrigger ActionTriggerType,
    tmxProgCompActionTime   DateAndTime,
    tmxProgCompMinBw        Integer32,
    tmxProgCompMaxBw        Integer32,
    tmxProgCompStatus       RowStatus
}
*****

tmxProgCompOpIndex OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This is used for unique identification and sequencing of all
        operational configuration messages. The value is the value of
        tmxOpNum when entry was created and is set by the manager."
    ::= { tmxProgCompEntry 1 }

```

```

tmxProgCompMsg      OBJECT-TYPE
    SYNTAX          BITS { create ( 0 ) , delete ( 1 ) }
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Each table row constitutes a command message. The message
        type indicates whether the Command type is to either create
        or delete the item described in this row. Any create command
        is further restricted by the Action Trigger Type. Depending on
        the Action Trigger Type, the command will either be immediately
        executed, executed at a time specified by Time of Action, or
        executed when a SPlceImmediateFlag is raised."
    ::= { tmxProgCompEntry 2 }

tmxProgCompTpeId     OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Identifier of the TPE to process this component,
        if any."
    ::= { tmxProgCompEntry 3 }

tmxProgCompSrcId     OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Physical input source identifier (ifIndex)"
    ::= { tmxProgCompEntry 4 }

tmxProgCompPID       OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "MPEG Input PID"
    ::= { tmxProgCompEntry 5 }

tmxProgCompAliasPID  OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "User assigned Component PID number in the output multiplex
        to create or delete."
    ::= { tmxProgCompEntry 6 }

tmxProgCompStreamType OBJECT-TYPE
    SYNTAX          BITS { video ( 0 ) , audio ( 1 ) , data ( 2 ) }
    MAX-ACCESS      read-create
    STATUS          current

```

## DESCRIPTION

"Stream content type of the MPEG component to be created or deleted."

::= { tmxProgCompEntry 7 }

tmxProgCompPriority OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-create

STATUS current

## DESCRIPTION

"This is a value between 1 (lowest) and 10 and indicates the minimum video quality (quantization level) after which opportunistic data can be added to the statistical multiplex group. Alternatively, if data within the group is guaranteed the this indicates the relative priority of the video within the group."

::= { tmxProgCompEntry 8 }

tmxProgCompTrcMode OBJECT-TYPE

SYNTAX BITS { transcode ( 0 ) , passthru ( 1 ) }

MAX-ACCESS read-create

STATUS current

## DESCRIPTION

"Transcode or passthrough this component."

::= { tmxProgCompEntry 9 }

tmxProgCompEmbeddedPCR OBJECT-TYPE

SYNTAX BITS { enable ( 0 ) , disable ( 1 ) }

MAX-ACCESS read-create

STATUS current

DESCRIPTION "PCR mode."

::= { tmxProgCompEntry 10 }

tmxProgCompActionTrigger OBJECT-TYPE

SYNTAX ActionTriggerType

MAX-ACCESS read-create

STATUS current

## DESCRIPTION

"Trigger type that will initiate this operation."

::= { tmxProgCompEntry 11 }

tmxProgCompActionTime OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-create

STATUS current

## DESCRIPTION

"If trigger type is a time of action this object contains the time."

::= { tmxProgCompEntry 12 }

tmxProgCompMinBw OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-create

```

STATUS          current
DESCRIPTION
    "Minimum bandwidth for program component.
    If minimum = maximum the bandwidth is fixed, i.e. not
    statistically multiplexed."
::= { tmxProgCompEntry 13 }

tmxProgCompMaxBw OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Maximum bandwidth for program component.
        If minimum = maximum the bandwidth is fixed, i.e. not
        statistically multiplexed."
    ::= { tmxProgCompEntry 14 }

tmxProgCompStatus OBJECT-TYPE
    SYNTAX          RowStatus
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Status of the row."
    ::= { tmxProgCompEntry 15 }
*****
tmxIPDataTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF TmxIPDataEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The IP Data Component Control table. Each row in this
        table represents a create/delete Op command. All
        Ops with the same Time of Action are to be executed
        in the order specified by the message index."
    ::= { tmxMediaControl 11 }

tmxIPDataEntry OBJECT-TYPE
    SYNTAX          TmxIPDataEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Entry"
    INDEX          { ifIndex, tmxStatGroupOpIndex, tmxProgOpIndex,
tmxIPDataOpIndex }
    ::= { tmxIPDataTable 1 }

TmxIPDataEntry ::= SEQUENCE {
    tmxIPDataOpIndex          Integer32,
    tmxIPDataMsg              BITS,
    tmxIPDataPID              Integer32,
    tmxIPDataDestAddr         IPAddress,
    tmxIPDataFlowCtlType      BITS,
    tmxIPDataTmxFlowCtlServerAddr IPAddress,
    tmxIPDataTmxFlowCtlServerPort Integer32,
    tmxIPDataTmxFlowCtlAddr   IPAddress,
    tmxIPDataTmxFlowCtlPort   Integer32,
    tmxIPDataMuxMode          BITS,

```

```

tmxIPDataOutMode          BITS,
tmxIPDataAvgBitRate       Integer32,
tmxIPDataAvgTimePeriod   Integer32,
tmxIPDataMinBitRate      Integer32,
tmxIPDataMaxBitRate      Integer32,
tmxIPDataActionTrigger   ActionTriggerType,
tmxIPDataActionTime      DateAndTime,
tmxIPDataSectionLength   BITS,
tmxIPDataStatus          RowStatus
)

```

\*\*\*\*\*

```

tmxIPDataOpIndex  OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This is used for unique identification and sequencing of all
        operational configuration messages. The value is the value of
        tmxOpNum when entry was created and is set by the manager."
    ::= { tmxIPDataEntry 1 }

```

```

tmxIPDataMsg      OBJECT-TYPE
    SYNTAX          BITS ( create ( 0 ) , delete ( 1 ) )
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "Each table row constitutes a command message. The message
        type indicates whether the Command type is to either create
        or delete the item described in this row. Any create command
        is further restricted by the Action Trigger Type. Depending on
        the Action Trigger Type, the command will either be immediately
        executed, executed at a time specified by Time of Action, or
        executed when a SPliceImmediateFlag is raised."
    ::= { tmxIPDataEntry 2 }

```

```

tmxIPDataPID      OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "MPEG PID number in output transport for this data stream.
        There may be multiple IP data route entries for the same PID!!!"
    ::= { tmxIPDataEntry 3 }

```

```

tmxIPDataDestAddr OBJECT-TYPE
    SYNTAX          IpAddress
    MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
        "IP address of destination host or network that this PID will
        route data for."
    ::= { tmxIPDataEntry 4 }

```

```

tmxIPDataFlowCtlType OBJECT-TYPE

```

```

SYNTAX          BITS { none ( 0 ) , smpte-325m ( 1 ) }
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "The type of flow control used by the TMX to throttle the data
server"
::= { tmxIPDataEntry 5 }

tmxIPDataTmxFlowCtlServerAddr OBJECT-TYPE
SYNTAX          IPAddress
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Server IP Address for flow control of this data stream."
::= { tmxIPDataEntry 6 }

tmxIPDataTmxFlowCtlServerPort OBJECT-TYPE
SYNTAX          Integer32
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Port number of the server's IP port for flow control of this data
stream."
::= { tmxIPDataEntry 7 }

tmxIPDataTmxFlowCtlAddr OBJECT-TYPE
SYNTAX          IPAddress
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "TMX IP Address for flow control of this data stream."
::= { tmxIPDataEntry 8 }

tmxIPDataTmxFlowCtlPort OBJECT-TYPE
SYNTAX          Integer32
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Port number of TMX's IP flow control port for this data stream."
::= { tmxIPDataEntry 9 }

tmxIPDataMuxMode OBJECT-TYPE
SYNTAX          BITS { guaranteed ( 0 ) , opportunistic ( 1 ) }
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Data multiplexing mode. Either guaranteed, in which case either
sufficiently much multiplexing bandwidth is allocated or the
video quality will be sacrificed in a statistical multiplex;
or opportunistic in which case data multiplexing is dependent
on spare bandwidth after desired video quality is achieved"
::= { tmxIPDataEntry 10 }

```

```

tmxIPDataOutMode      OBJECT-TYPE
    SYNTAX              BITS { atsc( 0 ) , dvb( 1 ) , other( 2 ) }
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
        "IP data encapsulation."
    ::= { tmxIPDataEntry 11 }

tmxIPDataAvgBitRate    OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
        "Average data bit rate over tmxIPDataAvgTimePeriod, if any."
    ::= { tmxIPDataEntry 12 }

tmxIPDataAvgTimePeriod OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
        "Averaging time period in Seconds, if any."
    ::= { tmxIPDataEntry 13 }

tmxIPDataMinBitRate    OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
        "Minimum guaranteed or desired bit rate over
tmxIPDataAvgTimePeriod,
        if any."
    ::= { tmxIPDataEntry 14 }

tmxIPDataMaxBitRate    OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
        "Maximum guaranteed or desired bit rate over
tmxIPDataAvgTimePeriod,
        if any."
    ::= { tmxIPDataEntry 15 }

tmxIPDataActionTrigger OBJECT-TYPE
    SYNTAX              ActionTriggerType
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
        "Trigger type that will initiate this operation."
    ::= { tmxIPDataEntry 16 }

```

```

tmxIPDataActionTime      OBJECT-TYPE
    SYNTAX                DateAndTime
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "If trigger type is a time of action this object contains the
time."
    ::= { tmxIPDataEntry 17 }

tmxIPDataSectionLength   OBJECT-TYPE
    SYNTAX                BITS { len4096(0), len1024(1) }
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION            "Maximum length of DSM-CC sections."
    ::= { tmxIPDataEntry 18 }

tmxIPDataStatus          OBJECT-TYPE
    SYNTAX                RowStatus
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION            "Status of the row."
    ::= { tmxIPDataEntry 19 }

*****
tmxAsyncDataTable        OBJECT-TYPE
    SYNTAX                SEQUENCE OF TmxAsyncDataEntry
    MAX-ACCESS            not-accessible
    STATUS                current
    DESCRIPTION
        "The RS-232/RS-422 Data Component Control table. Each row in this
table represents a create/delete Op command. All
Ops with the same Time of Action are to be executed
in the order specified by the message index."
    ::= { tmxMediaControl 12 }

tmxAsyncDataEntry        OBJECT-TYPE
    SYNTAX                TmxAsyncDataEntry
    MAX-ACCESS            not-accessible
    STATUS                current
    DESCRIPTION            "Entry"
    INDEX                 { ifIndex, tmxStatGroupOpIndex, tmxProgOpIndex ,
tmxAsyncDataOpIndex }
    ::= { tmxAsyncDataTable 1 }

TmxAsyncDataEntry ::= SEQUENCE {
    tmxAsyncDataOpIndex      Integer32,
    tmxAsyncDataMsg          BITS,
    tmxAsyncDataSrcId        Integer32,
    tmxAsyncDataPID          Integer32,
    tmxAsyncDataBaudRate     Integer32,
    tmxAsyncDataParity       Integer32,
    tmxAsyncDataBits         Integer32,
    tmxAsyncDataStopBits     Integer32,
    tmxAsyncDataFlowControl  BITS,

```



```

tmxAsyncDataMuxMode          BITS,
tmxAsyncDataOutMode          BITS,
tmxAsyncDataAvgBitRate       Integer32,
tmxAsyncDataAvgTimePeriod    Integer32,
tmxAsyncDataMinBitRate       Integer32,
tmxAsyncDataMaxBitRate       Integer32,
tmxAsyncDataActionTrigger    ActionTriggerType,
tmxAsyncDataActionTime       DateAndTime,
tmxAsyncDataStatus           RowStatus
}

```

\*\*\*\*\*

```

tmxAsyncDataOpIndex          OBJECT-TYPE
    SYNTAX                    Integer32
    MAX-ACCESS                not-accessible
    STATUS                    current
    DESCRIPTION
        "This is used for unique identification and sequencing of all
        operational configuration messages. The value is the value of
        tmxOpNum when entry was created and is set by the manager."
    ::= { tmxAsyncDataEntry 1 }

```

```

tmxAsyncDataMsg              OBJECT-TYPE
    SYNTAX                    BITS { create ( 0 ) , delete ( 1 ) }
    MAX-ACCESS                read-create
    STATUS                    current
    DESCRIPTION
        "Each table row constitutes a command message. The message
        type indicates whether the Command type is to either create
        or delete the item described in this row. Any create command
        is further restricted by the Action Trigger Type. Depending on
        the Action Trigger Type, the command will either be immediately
        executed, executed at a time specified by Time of Action, or
        executed when a SPliceImmediateFlag is raised."
    ::= { tmxAsyncDataEntry 2 }

```

```

tmxAsyncDataSrcId            OBJECT-TYPE
    SYNTAX                    Integer32
    MAX-ACCESS                read-create
    STATUS                    current
    DESCRIPTION
        "Physical input identifier (ifIndex)."
    ::= { tmxAsyncDataEntry 3 }

```

```

tmxAsyncDataPID              OBJECT-TYPE
    SYNTAX                    Integer32
    MAX-ACCESS                read-create
    STATUS                    current
    DESCRIPTION
        "MPEG PID number in output multiplex for this data stream."
    ::= { tmxAsyncDataEntry 4 }

```

```

tmxAsyncDataBaudRate         OBJECT-TYPE
    SYNTAX                    Integer32
    MAX-ACCESS                read-create

```

```

STATUS          current
DESCRIPTION
    "RS-232 baud rate setting, if any."
::= { tmxAsyncDataEntry 5 }

```

```

tmxAsyncDataParity    OBJECT-TYPE
SYNTAX               Integer32
MAX-ACCESS            read-create
STATUS                current
DESCRIPTION
    "RS-232 parity, if any."
::= { tmxAsyncDataEntry 6 }

```

```

tmxAsyncDataBits      OBJECT-TYPE
SYNTAX               Integer32
MAX-ACCESS            read-create
STATUS                current
DESCRIPTION
    "RS-232 number of data bits, if any."
::= { tmxAsyncDataEntry 7 }

```

```

tmxAsyncDataStopBits  OBJECT-TYPE
SYNTAX               Integer32
MAX-ACCESS            read-create
STATUS                current
DESCRIPTION
    "RS-232, number of stop bits, if any."
::= { tmxAsyncDataEntry 8 }

```

```

tmxAsyncDataFlowControl OBJECT-TYPE
SYNTAX               BITS { xonxoff ( 0 ) , hw ( 1 ) , none ( 2 ) }
MAX-ACCESS            read-create
STATUS                current
DESCRIPTION
    "RS-232 flow control, if any."
::= { tmxAsyncDataEntry 9 }

```

```

tmxAsyncDataMuxMode   OBJECT-TYPE
SYNTAX               BITS { guaranteed ( 0 ) , opportunistic ( 1 ) }
MAX-ACCESS            read-create
STATUS                current
DESCRIPTION
    "Data multiplexing mode. Either guaranteed, in which case either
    sufficiently much multiplexing bandwidth is allocated or the
    video quality will be sacrificed in a statistical multiplex;
    or opportunistic in which case data multiplexing is dependent
    on spare bandwidth after desired video quality is achieved"
::= { tmxAsyncDataEntry 10 }

```

```

tmxAsyncDataOutMode   OBJECT-TYPE
SYNTAX               BITS { atsc ( 0 ) , dvb ( 1 ) }
MAX-ACCESS            read-create
STATUS                current

```

```

DESCRIPTION
    "Data encapsulation on output."
 ::= { tmxAsyncDataEntry 11 }

tmxAsyncDataAvgBitRate      OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS   read-create
    STATUS      current
    DESCRIPTION
        "Average data bit rate over tmxAsyncDataAvgTimePeriod, if
any."
 ::= { tmxAsyncDataEntry 12 }

tmxAsyncDataAvgTimePeriod  OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS   read-create
    STATUS      current
    DESCRIPTION
        "Averaging time period in Seconds, if any."
 ::= { tmxAsyncDataEntry 13 }

tmxAsyncDataMinBitRate     OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS   read-create
    STATUS      current
    DESCRIPTION
        "Minimum guaranteed or desired bit rate over
        tmxAsyncDataAvgTimePeriod, if any."
 ::= { tmxAsyncDataEntry 14 }

tmxAsyncDataMaxBitRate     OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS   read-create
    STATUS      current
    DESCRIPTION
        "Maximum guaranteed or desired bit rate over
        tmxAsyncDataAvgTimePeriod, if any."
 ::= { tmxAsyncDataEntry 15 }

tmxAsyncDataActionTrigger  OBJECT-TYPE
    SYNTAX      ActionTriggerType
    MAX-ACCESS   read-create
    STATUS      current
    DESCRIPTION
        "Trigger type that will initiate this operation."
 ::= { tmxAsyncDataEntry 16 }

tmxAsyncDataActionTime     OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS   read-create
    STATUS      current
    DESCRIPTION

```

the time." "If trigger type is a time of action this object contains

```
::= { tmxAsyncDataEntry 17 }
```

```
tmxAsyncDataStatus      OBJECT-TYPE
    SYNTAX               RowStatus
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "Status of the row."
    ::= { tmxAsyncDataEntry 18 }
```

\*\*\*\*\*

```
tmxCarouselTable SEQUENCE OF OBJECT-TYPE
    SYNTAX               SEQUENCE OF TmxCarouselEntry
    MAX-ACCESS            not-accessible
    STATUS                current
    DESCRIPTION
        "The Carousel Control table. Each row in this
        table represents a create/delete Op command. All
        Ops with the same Time of Action are to be executed
        in the order specified by the message index."
    ::= { tmxMediaControl 13 }
```

```
tmxCarouselEntry        OBJECT-TYPE
    SYNTAX               TmxCarouselEntry
    MAX-ACCESS            not-accessible
    STATUS                current
    DESCRIPTION
        "Entry"
    INDEX                { ifIndex, tmxStatGroupOpIndex, tmxProgOpIndex ,
tmxCarouselOpIndex }
    ::= { tmxCarouselTable 1 }
```

```
TmxCarouselEntry ::= SEQUENCE {
    tmxCarouselOpIndex      Integer32,
    tmxCarouselMsg          BITS,
    tmxCarouselPID          Integer32,
    tmxCarouselOutStandard  BITS,
    tmxCarouselRepetitionRate Integer32,
    tmxCarouselLoadData     OCTET STRING,
    tmxCarouselActionTrigger ActionType,
    tmxCarouselActionTime   DateAndTime,
    tmxCarouselOnDemand     BITS,
    tmxCarouselStatus       RowStatus
}
```

\*\*\*\*\*

```
tmxCarouselOpIndex      OBJECT-TYPE
    SYNTAX               Integer32
    MAX-ACCESS            not-accessible
    STATUS                current
    DESCRIPTION
        "This is used for unique identification and sequencing of all
        operational configuration messages. The value is the value of
        tmxOpNum when entry was created and is set by the manager."
    ::= { tmxCarouselEntry 1 }
```

```

tmxCarouselMsg          OBJECT-TYPE
    SYNTAX               BITS { create ( 0 ) , delete ( 1 ) }
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "Each table row constitutes a command message. The message
        type indicates whether the Command type is to either create
        or delete the item described in this row. Any create command
        is further restricted by the Action Trigger Type. Depending on
        the Action Trigger Type, the command will either be immediately
        executed, executed at a time specified by Time of Action, or
        executed when a SPliceImmediateFlag is raised."
 ::= { tmxCarouselEntry 2 }

tmxCarouselPID          OBJECT-TYPE
    SYNTAX               Integer32
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "MPEG PID number in output multiplex."
 ::= { tmxCarouselEntry 3 }

tmxCarouselOutStandard  OBJECT-TYPE
    SYNTAX               BITS { atsc ( 0 ) , dvb ( 1 ) }
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "Data encapsulation on output, if any."
 ::= { tmxCarouselEntry 4 }

tmxCarouselRepetitionRate OBJECT-TYPE
    SYNTAX               Integer32
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "Repetition rate of the data in milliseconds."
 ::= { tmxCarouselEntry 5 }

tmxCarouselLoadData     OBJECT-TYPE
    SYNTAX               OCTET STRING
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "File name on server or if LoadHost=NULL, this
        is the data itself that is to be put onto the
        Carousel."
 ::= { tmxCarouselEntry 6 }

tmxCarouselActionTrigger OBJECT-TYPE
    SYNTAX               ActionTriggerType
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "Trigger type that will initiate this operation."

```

```

::= { tmxCarouselEntry 7 }

tmxCarouselActionTime      OBJECT-TYPE
    SYNTAX                 DateAndTime
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION             "If trigger type is a time of action this object contains
the time."
::= { tmxCarouselEntry 8 }

tmxCarouselOnDemand        OBJECT-TYPE
    SYNTAX                 BITS { disabled(0), enabled(1) }
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION             "Switch for on-demand table generation"
::= { tmxCarouselEntry 9 }

tmxCarouselStatus         OBJECT-TYPE
    SYNTAX                 RowStatus
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION             "Status of the row."
::= { tmxCarouselEntry 10 }

*****
tmxFifoTable              OBJECT-TYPE
    SYNTAX                 SEQUENCE OF TmxFifoEntry
    MAX-ACCESS              not-accessible
    STATUS                  current
    DESCRIPTION             "The FIFO Control table. Each row in this
                             table represents a create/delete Op command. All
                             Ops with the same Time of Action are to be executed
                             in the order specified by the message index."
::= { tmxMediaControl 14 }

tmxFifoEntry              OBJECT-TYPE
    SYNTAX                 TmxFifoEntry
    MAX-ACCESS              not-accessible
    STATUS                  current
    DESCRIPTION             "Entry"
    INDEX                  { ifIndex, tmxStatGroupOpIndex, tmxProgOpIndex ,
tmxFifoOpIndex }
::= { tmxFifoTable 1 }

TmxFifoEntry ::= SEQUENCE {
    tmxFifoOpIndex          Integer32,
    tmxFifoMsg              BITS,
    tmxFifoPID              Integer32,
    tmxFifoOutStandard      BITS,
    tmxFifoAvgBitRate        Integer32,
    tmxFifoAvgTimePeriod    Integer32,
    tmxFifoMinBitRate        Integer32,
    tmxFifoMaxBitRate        Integer32,
    tmxFifoLoadTime          Integer32,

```

```

tmxFifoLoadType          BITS,
tmxFifoLoadHost          IPAddress,
tmxFifoFileName          DisplayString,
tmxFifoLoadDataType      BITS,
tmxFifoActionTrigger     ActionTriggerType,
tmxFifoActionTime        DateAndTime,
tmxFifoOnDemand          BITS,
tmxFifoStatus            RowStatus
}
*****

tmxFifoOpIndex           OBJECT-TYPE
    SYNTAX                Integer32
    MAX-ACCESS            not-accessible
    STATUS                current
    DESCRIPTION
        "This is used for unique identification and sequencing of all
        operational configuration messages. The value is the value of
        tmxOpNum when entry was created and is set by the manager."
    ::= { tmxFifoEntry 1 }

tmxFifoMsg               OBJECT-TYPE
    SYNTAX                BITS { create ( 0 ) , delete ( 1 ) }
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "Each table row constitutes a command message. The message
        type indicates whether the Command type is to either create
        or delete the item described in this row. Any create command
        is further restricted by the Action Trigger Type. Depending on
        the Action Trigger Type, the command will either be immediately
        executed, executed at a time specified by Time of Action, or
        executed when a SPliceImmediateFlag is raised."
    ::= { tmxFifoEntry 2 }

tmxFifoPID               OBJECT-TYPE
    SYNTAX                Integer32
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "MPEG PID number in output multiplex."
    ::= { tmxFifoEntry 3 }

tmxFifoOutStandard       OBJECT-TYPE
    SYNTAX                BITS { atsc ( 0 ) , dvb ( 1 ) }
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION
        "Data encapsulation on output, if any."
    ::= { tmxFifoEntry 4 }

tmxFifoAvgBitRate        OBJECT-TYPE
    SYNTAX                Integer32
    MAX-ACCESS            read-create
    STATUS                current
    DESCRIPTION

```

```

                                "Average data bit rate over tmxFifoAvgTimePeriod, if any."
 ::= { tmxFifoEntry 5 }

tmxFifoAvgTimePeriod          OBJECT-TYPE
    SYNTAX                     Integer32
    MAX-ACCESS                 read-create
    STATUS                     current
    DESCRIPTION
        "Averaging time period in Seconds, if any."
 ::= { tmxFifoEntry 6 }

tmxFifoMinBitRate             OBJECT-TYPE
    SYNTAX                     Integer32
    MAX-ACCESS                 read-create
    STATUS                     current
    DESCRIPTION
        "Minimum guaranteed or desired bit rate over
         tmxFifoAvgTimePeriod, if any."
 ::= { tmxFifoEntry 7 }

tmxFifoMaxBitRate             OBJECT-TYPE
    SYNTAX                     Integer32
    MAX-ACCESS                 read-create
    STATUS                     current
    DESCRIPTION
        "Maximum guaranteed or desired bit rate over
         tmxFifoAvgTimePeriod, if any."
 ::= { tmxFifoEntry 8 }

tmxFifoLoadTime               OBJECT-TYPE
    SYNTAX                     Integer32
    MAX-ACCESS                 read-create
    STATUS                     current
    DESCRIPTION
        "Time at which to load the file from server, if any."
 ::= { tmxFifoEntry 9 }

tmxFifoLoadType               OBJECT-TYPE
    SYNTAX                     BITS { tftp ( 0 ) , ftp ( 1 ) , nfs ( 2 ) , snmpdata (
3 ) }
    MAX-ACCESS                 read-create
    STATUS                     current
    DESCRIPTION
        "FIFO load protocol."
 ::= { tmxFifoEntry 10 }

tmxFifoLoadHost               OBJECT-TYPE
    SYNTAX                     IpAddress
    MAX-ACCESS                 read-create
    STATUS                     current
    DESCRIPTION
        "IP address of server."
 ::= { tmxFifoEntry 11 }

```



```

tmxFifoFileName      OBJECT-TYPE
    SYNTAX            DisplayString
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "File name on server."
    ::= { tmxFifoEntry 12 }

tmxFifoLoadDataType   OBJECT-TYPE
    SYNTAX            BITS { data ( 0 ) , dsmcc ( 1 ) , prepacketized ( 2
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "Format of data to be put onto the FIFO."
    ::= { tmxFifoEntry 13 }

tmxFifoActionTrigger  OBJECT-TYPE
    SYNTAX            ActionTriggerType
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "Trigger type that will initiate this operation."
    ::= { tmxFifoEntry 14 }

tmxFifoActionTime     OBJECT-TYPE
    SYNTAX            DateAndTime
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "If trigger type is a time of action this object contains the
time."
    ::= { tmxFifoEntry 15 }

tmxFifoOnDemand       OBJECT-TYPE
    SYNTAX            BITS { disabled(0), enabled(1) }
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "Switch for on-demand table generation"
    ::= { tmxFifoEntry 16 }

tmxFifoStatus         OBJECT-TYPE
    SYNTAX            RowStatus
    MAX-ACCESS        read-create
    STATUS            current
    DESCRIPTION
        "Status of the row."
    ::= { tmxFifoEntry 17 }
*****
tmxFifoProgInsertTable OBJECT-TYPE
    SYNTAX            SEQUENCE OF TmxFifoProgInsertEntry
    MAX-ACCESS        not-accessible
    STATUS            current
    DESCRIPTION
        "Table to control TMX's digital program insertion (Ads, NVOD)

```

and splicing. Each row in this table represents a create/delete Op command. All Ops with the same Time of Action are to be executed in the order specified by the message index."

```

::= { tmxMediaControl 15 }

tmxProgInsertEntry          OBJECT-TYPE
    SYNTAX                  TmxProgInsertEntry
    MAX-ACCESS              not-accessible
    STATUS                  current
    DESCRIPTION
        "Entry"
    INDEX                   { ifIndex, tmxStatGroupOpIndex, tmxProgOpIndex,
tmxProgInsertOpIndex }
    ::= { tmxProgInsertTable 1 }

TmxProgInsertEntry ::= SEQUENCE {
    tmxProgInsertOpIndex      Integer32,
    tmxProgInsertMsg          BITS,
    tmxProgInsertProgName     DisplayString,
    tmxProgInsertSplicerName  DisplayString,
    tmxProgInsertSplicerType  BITS,
    tmxProgInsertServerAddr   IpAddress,
    tmxProgInsertServerPort   Integer32,
    tmxProgInsertAdSrcId      Integer32,
    tmxProgInsertTmxAddr      IpAddress,
    tmxProgInsertTmxPort      Integer32,
    tmxProgInsertActionTrigger ActionType,
    tmxProgInsertActionTime   DateAndTime,
    tmxProgInsertStatus       RowStatus
}
*****

tmxProgInsertOpIndex        OBJECT-TYPE
    SYNTAX                  Integer32
    MAX-ACCESS              not-accessible
    STATUS                  current
    DESCRIPTION
        "This is used for unique identification and sequencing of all
        operational configuration messages. The value is the value of
        tmxOpNum when entry was created and is set by the manager."
    ::= { tmxProgInsertEntry 1 }

tmxProgInsertMsg            OBJECT-TYPE
    SYNTAX                  BITS { create ( 0 ) , delete ( 1 ) }
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION
        "Each table row constitutes a command message. The message
        type indicates whether the Command type is to either create
        or delete the item described in this row. Any create command
        is further restricted by the Action Trigger Type. Depending on
        the Action Trigger Type, the command will either be immediately
        executed, executed at a time specified by Time of Action, or
        executed when a SPliceImmediateFlag is raised."
    ::= { tmxProgInsertEntry 2 }

```

```

tmxProgInsertProgName      OBJECT-TYPE
    SYNTAX                  DisplayString
    MAX-ACCESS               read-create
    STATUS                   current
    DESCRIPTION              "User assigned program name of the program being spliced
into."
    ::= { tmxProgInsertEntry 3 }

tmxProgInsertSplicerName   OBJECT-TYPE
    SYNTAX                  DisplayString
    MAX-ACCESS               read-create
    STATUS                   current
    DESCRIPTION              "Name of the splicer if more then one in TMX (as in DVS
standard)."
    ::= { tmxProgInsertEntry 4 }

tmxProgInsertSplicerType   OBJECT-TYPE
    SYNTAX                  BITS { adstreaminsert (0) , nvodstreaminsert (1),
                                actvstreaminsert(2) }
    MAX-ACCESS               read-create
    STATUS                   current
    DESCRIPTION              "Type of splicer."
    ::= { tmxProgInsertEntry 5 }

tmxProgInsertServerAddr    OBJECT-TYPE
    SYNTAX                  IPAddress
    MAX-ACCESS               read-create
    STATUS                   current
    DESCRIPTION              "IP address of server for this stream insertion for
DVS 380 API."
    ::= { tmxProgInsertEntry 6 }

tmxProgInsertServerPort    OBJECT-TYPE
    SYNTAX                  Integer32
    MAX-ACCESS               read-create
    STATUS                   current
    DESCRIPTION              "Server TCP port number for control of this stream insertion
for DVS 380 API."
    ::= { tmxProgInsertEntry 7 }

tmxProgInsertAdSrcId       OBJECT-TYPE
    SYNTAX                  Integer32
    MAX-ACCESS               read-create
    STATUS                   current
    DESCRIPTION              "Physical input identifier (ifIndex) of the ad/NVOD stream."
    ::= { tmxProgInsertEntry 8 }

tmxProgInsertTmxAddr       OBJECT-TYPE

```

66

```

SYNTAX      IPAddress
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION  "IP address of TMX for this stream insertion for DVS 380
API."
 ::= { tmxProgInsertEntry 9 }

tmxProgInsertTmxPort OBJECT-TYPE
SYNTAX      Integer32
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION  "TMX TCP port number for control of this stream insertion
              for DVS 380 API."
 ::= { tmxProgInsertEntry 10 }

tmxProgInsertActionTrigger OBJECT-TYPE
SYNTAX      ActionType
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION  "Trigger type that will initiate this operation."
 ::= { tmxProgInsertEntry 11 }

tmxProgInsertActionTime OBJECT-TYPE
SYNTAX      DateAndTime
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION  "If trigger type is a time of action this object contains the
time."
 ::= { tmxProgInsertEntry 12 }

tmxProgInsertStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION  "Status of the row."
 ::= { tmxProgInsertEntry 13 }

-- tmxMediaMonitor tables enable monitoring of MPEG input/output information.
-- They consist of:
--     tmxInputPIDTable for input rate monitoring per PID
--     tmxOutputPIDTable for output rate monitoring per PID
--     tmxPisiTable for MPEG Table monitoring per PID
*****
tmxInputPIDTable OBJECT-TYPE
SYNTAX      SEQUENCE OF TmxInputPIDEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION  "Table with input PID statistics."
 ::= { tmxMediaMonitor 1 }

```

```

tmxInputPIDEntry      OBJECT-TYPE
    SYNTAX              TmxInputPIDEntry
    MAX-ACCESS          not-accessible
    STATUS              current
    DESCRIPTION
        "Entry"
    INDEX               ( ifIndex, tmxInputPIDId )
    ::= { tmxInputPIDTable 1 }

TmxInputPIDEntry ::= SEQUENCE {
    tmxInputPIDId      Integer32,
    tmxInputPIDBitRate Integer32,
    tmxInputPIDEntryStatus RowStatus
}
*****
tmxInputPIDId      OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Component PID with stream type."
    ::= { tmxInputPIDEntry 1 }

tmxInputPIDBitRate      OBJECT-TYPE
    SYNTAX              Integer32
    MAX-ACCESS          read-only
    STATUS              current
    DESCRIPTION
        "Bit rate in bps."
    ::= { tmxInputPIDEntry 2 }

tmxInputPIDEntryStatus      OBJECT-TYPE
    SYNTAX              RowStatus
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
        "Status of this entry."
    ::= { tmxInputPIDEntry 3 }
*****
tmxOutputPIDTable      OBJECT-TYPE
    SYNTAX              SEQUENCE OF TmxOutputPIDEntry
    MAX-ACCESS          not-accessible
    STATUS              current
    DESCRIPTION
        "Table with output PID statistics."
    ::= { tmxMediaMonitor 2 }

tmxOutputPIDEntry      OBJECT-TYPE
    SYNTAX              TmxOutputPIDEntry
    MAX-ACCESS          not-accessible
    STATUS              current
    DESCRIPTION
        "Entry"
    INDEX               ( ifIndex, tmxOutputPIDId )
    ::= { tmxOutputPIDTable 1 }

TmxOutputPIDEntry ::= SEQUENCE {

```

tmxOutputPIDId	Integer32,
tmxOutputPIDBitRate	Integer32,
tmxOutputPIDMinBitRate	Integer32,
tmxOutputPIDMaxBitRate	Integer32,
tmxOutputPIDFrameRate	Integer32,
tmxOutputPIDResolution	Integer32,
tmxOutputPIDBFrames	Integer32,
tmxOutputPIDFilmMode	Integer32,
tmxOutputPIDEntryStatus	RowStatus
}	

\*\*\*\*\*

tmxOutputPIDId	OBJECT-TYPE
SYNTAX	Integer32
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"Component PID with stream type."
::= {	tmxOutputPIDEntry 1 }

tmxOutputPIDBitRate	OBJECT-TYPE
SYNTAX	Integer32
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"Bit rate in bps."
::= {	tmxOutputPIDEntry 2 }

tmxOutputPIDMinBitRate	OBJECT-TYPE
SYNTAX	Integer32
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"Min Bit rate in bps."
::= {	tmxOutputPIDEntry 3 }

tmxOutputPIDMaxBitRate	OBJECT-TYPE
SYNTAX	Integer32
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"Max Bit rate in bps."
::= {	tmxOutputPIDEntry 4 }

tmxOutputPIDFrameRate	OBJECT-TYPE
SYNTAX	Integer32
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"Frame rate if the PID is a transcoded video PID."
::= {	tmxOutputPIDEntry 5 }

tmxOutputPIDResolution	OBJECT-TYPE
SYNTAX	Integer32

```

MAX-ACCESS      read-only
STATUS          current
DESCRIPTION     "Number of Macroblocks per Second if PID is a transcoded
video PID."
 ::= { tmxOutputPIDEntry 6 }

tmxOutputPIDBFrames OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Average ratio of B frames to I and P frames if PID is a
transcoded video PID."
 ::= { tmxOutputPIDEntry 7 }

tmxOutputPIDFilmMode OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Film mode status if PID is a transcoded video PID."
 ::= { tmxOutputPIDEntry 8 }

tmxOutputPIDEntryStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION "Status of this entry."
 ::= { tmxOutputPIDEntry 9 }

*****
tmxPsiSiTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF TmxPsiSiEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "Table of all input transport streams."
 ::= { tmxMediaMonitor 3 }

tmxPsiSiEntry OBJECT-TYPE
    SYNTAX      TmxPsiSiEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "Entry"
    INDEX      { ifIndex, tmxPsiSiPidNum, tmxPsiSiTableId,
tmxPsiSiCurrNext }
 ::= { tmxPsiSiTable 1 }

TmxPsiSiEntry ::= SEQUENCE {
    tmxPsiSiPidNum      Integer32,
    tmxPsiSiTableId     Integer32,
    tmxPsiSiCurrNext    Integer32,
    tmxPsiSiTableInfo   OCTET STRING,
    tmxPsiSiVersion     Integer32,

```

```

tmxPsiSiEntryStatus      RowStatus
}
*****

tmxPsiSiPidNum            OBJECT-TYPE
    SYNTAX                Integer32
    MAX-ACCESS             not-accessible
    STATUS                 current
    DESCRIPTION
        "Transport Stream PID number"
    ::= { tmxPsiSiEntry 1 }

tmxPsiSiTableId           OBJECT-TYPE
    SYNTAX                Integer32
    MAX-ACCESS             not-accessible
    STATUS                 current
    DESCRIPTION
        "Transport Stream Table Id"
    ::= { tmxPsiSiEntry 2 }

tmxPsiSiCurrNext          OBJECT-TYPE
    SYNTAX                Integer32
    MAX-ACCESS             not-accessible
    STATUS                 current
    DESCRIPTION
        "MPEG Table currnet next indicator."
    ::= { tmxPsiSiEntry 3 }

tmxPsiSiTableInfo         OBJECT-TYPE
    SYNTAX                OCTET STRING
    MAX-ACCESS             read-only
    STATUS                 current
    DESCRIPTION
        "Transport Stream Table."
    ::= { tmxPsiSiEntry 4 }

tmxPsiSiVersion           OBJECT-TYPE
    SYNTAX                Integer32
    MAX-ACCESS             read-only
    STATUS                 current
    DESCRIPTION
        "Table version for this transport stream"
    ::= { tmxPsiSiEntry 5 }

tmxPsiSiEntryStatus       OBJECT-TYPE
    SYNTAX                RowStatus
    MAX-ACCESS             read-create
    STATUS                 current
    DESCRIPTION
        "Status of this entry."
    ::= { tmxPsiSiEntry 6 }

tmxMuxMemoryTable         OBJECT-TYPE
    SYNTAX                SEQUENCE OF TmxMuxMemoryEntry
    MAX-ACCESS             not-accessible
    STATUS                 current
    DESCRIPTION
        "Table indicating the maximum contiguous block of memory available
for carousels."

```



```

::= { tmxMediaMonitor 4 }

tmxMuxMemoryEntry          OBJECT-TYPE
    SYNTAX                  TmxMuxMemoryEntry
    MAX-ACCESS              not-accessible
    STATUS                  current
    DESCRIPTION
        "Entry"
    INDEX                   {ifIndex}
    ::= { tmxMuxMemoryTable 1 }

TmxMuxMemoryEntry ::= SEQUENCE {
    tmxMuxMemorySize        Integer32,
    tmxMuxMemoryEntryStatus RowStatus
}

tmxMuxMemorySize           OBJECT-TYPE
    SYNTAX                  Integer32
    MAX-ACCESS              read-only
    STATUS                  current
    DESCRIPTION
        "The maximum byte size currently available."
    ::= { tmxMuxMemoryEntry 1 }

tmxMuxMemoryEntryStatus    OBJECT-TYPE
    SYNTAX                  RowStatus
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION
        "Status of this entry."
    ::= { tmxMuxMemoryEntry 2 }

```

END

**[0081]** While the present invention has been described in connection with what is presently considered to be the most practical and preferred embodiments, it is to be understood that the invention is not limited to the disclosed embodiments, but is intended to encompass the various modifications and equivalent arrangements included within the spirit and scope of the appended claims. With respect to the above description, for example, it is to be realized that the optimum implementation, function and manner of operation, assembly and use, are deemed readily apparent to one skilled in the art, and all equivalent relationships to those illustrated in the drawings and described in the specification are intended to be encompassed by the appended claims. Therefore, the foregoing is considered to be an illustrative, not exhaustive, description of the principles of the present invention.

What is claimed is:

1. A graphical user interface for use in a transport multiplexer having plural input ports and plural output ports wherein dynamic content streams are received at the input ports and routed to the output ports, the content streams comprising plural components and having attributes that include bit rate, type and name, the graphical user interface comprising:

a tree view screen for displaying graphical objects representing the input ports and the output ports and for displaying input and output trees with plural branches indicating the hierarchical structure of respective input and output content streams, the tree view screen indicating the content streams received at and routing through the transport multiplexer and displaying, in real-time, changes in the content streams received at the transport multiplexer.

2. The graphical user interface of claim 1 wherein at least some of the graphical objects are icons and wherein selection of an icon displays a transport editor window in which the content stream attributes may be modified.

3. The graphical user interface of claim 2 wherein output bandwidth utilization is recalculated and redisplayed in real-time in response to at least one content stream attribute being modified.

4. The graphical user interface of claim 1 wherein the tree view screen further comprises a scrolling log message window for displaying system generated log messages in real-time.

5. The graphical user interface of claim 4 wherein the log messages to be displayed are filtered in accordance with plural predetermined filter levels.

6. The graphical user interface of claim 1 wherein the trees further comprise graphical objects indicative of the type of content streams received by the input ports and exiting the output ports.
7. A graphical user interface for use in a transport multiplexer with a chassis having plural hardware and software components to route input content streams received by input ports to predetermined output ports, the graphical user interface comprising:
  - a chassis view screen for displaying graphical objects representing the physical structure of the hardware components installed in the chassis of the transport multiplexer, the graphical objects indicating the type of input and output ports provided by the hardware components.
8. The graphical user interface of claim 7 wherein the chassis view screen also displays in real-time the operational status of the displayed hardware components.
9. The graphical user interface of claim 7 wherein the graphical user interface further comprises a software attributes screen for displaying attributes of the software utilized in the transport multiplexer, the software attributes comprising software version number.
10. The graphical user interface of claim 7 wherein the interface is capable of graphically displaying in real-time changes detected in the hardware status data.
11. A method of using a graphical user interface to configure routing control of a content stream through a transport multiplexer, the router having plural input ports and plural output ports, the method comprising:
  - displaying icons representing the input ports;
  - displaying icons representing the output ports;

displaying an icon representing a input content stream, the stream being received by one of the input ports and the input stream icon being displayed in proximity to the input port icon;

dragging the input stream icon into proximity to an output port icon;

dropping the input content stream icon in proximity with the output port icon; and

responsive to the step of dropping, displaying an icon representing an output stream exiting the output port, the icon being displayed substantially where the input stream was dropped.

12. The method of claim 11 wherein the output stream icon represents an output stream presently exiting the output port.

13. The method of claim 11 wherein the output stream icon represents an output stream that will begin exiting the output port at a predetermined time in the future.

14. The method of claim 11, wherein:

the input content stream comprises plural hierarchical components having attributes,

the input stream icon represents all of the plural hierarchical components, and

the output stream icon represents a content stream having the same plural hierarchical components as the input stream.

15. The method of claim 11 further comprising:

displaying an alphanumeric character string in proximity with the input stream icon;

dragging the alphanumeric character string with the input stream icon;

dropping the alphanumeric character string with the input stream icon; and

displaying an alphanumeric character string in proximity with the output stream icon.

16. A graphical user interface for use in a transport multiplexer having plural input ports and plural output ports wherein content streams are received at the input ports and routed to the output ports, the content streams comprising a hierarchy of plural components and having attributes that include bit rate, type and name, the graphical user interface comprising:

a tree view screen for displaying icons representing the input ports and the output ports and input and output trees with plural branches indicating the hierarchical structure of the input and output content streams;

the graphical user interface permitting configuration of routing controls by dragging content stream components from the input tree and dropping the content stream components onto the output tree whereby the content stream is automatically routed through the transport multiplexer to a desired output.

17. The graphical user interface of claim 16 wherein plural content stream components can be simultaneously dragged from the input tree and dropped into the output tree whereby plural content streams are automatically routed through the transport multiplexer to the output.

18. A graphical user interface for use in a transport multiplexer having plural input ports and plural output ports wherein dynamic content streams are received at the input ports and routed to the output ports in accordance with routing control data, the graphical user interface comprising:

plural tree view screens for displaying icons representing the input ports and the output ports, the tree view screens also displaying input and output trees with plural branches indicating the hierarchical structure of respective input and output content streams, each of the tree view screens showing content stream routing for a unique time segment;

a time bar comprising plural time segments defined between pairs of segment start and stop times; and

the graphical user interface permitting consecutive display of the plural tree view screens by selection of corresponding time segments of the time bar.

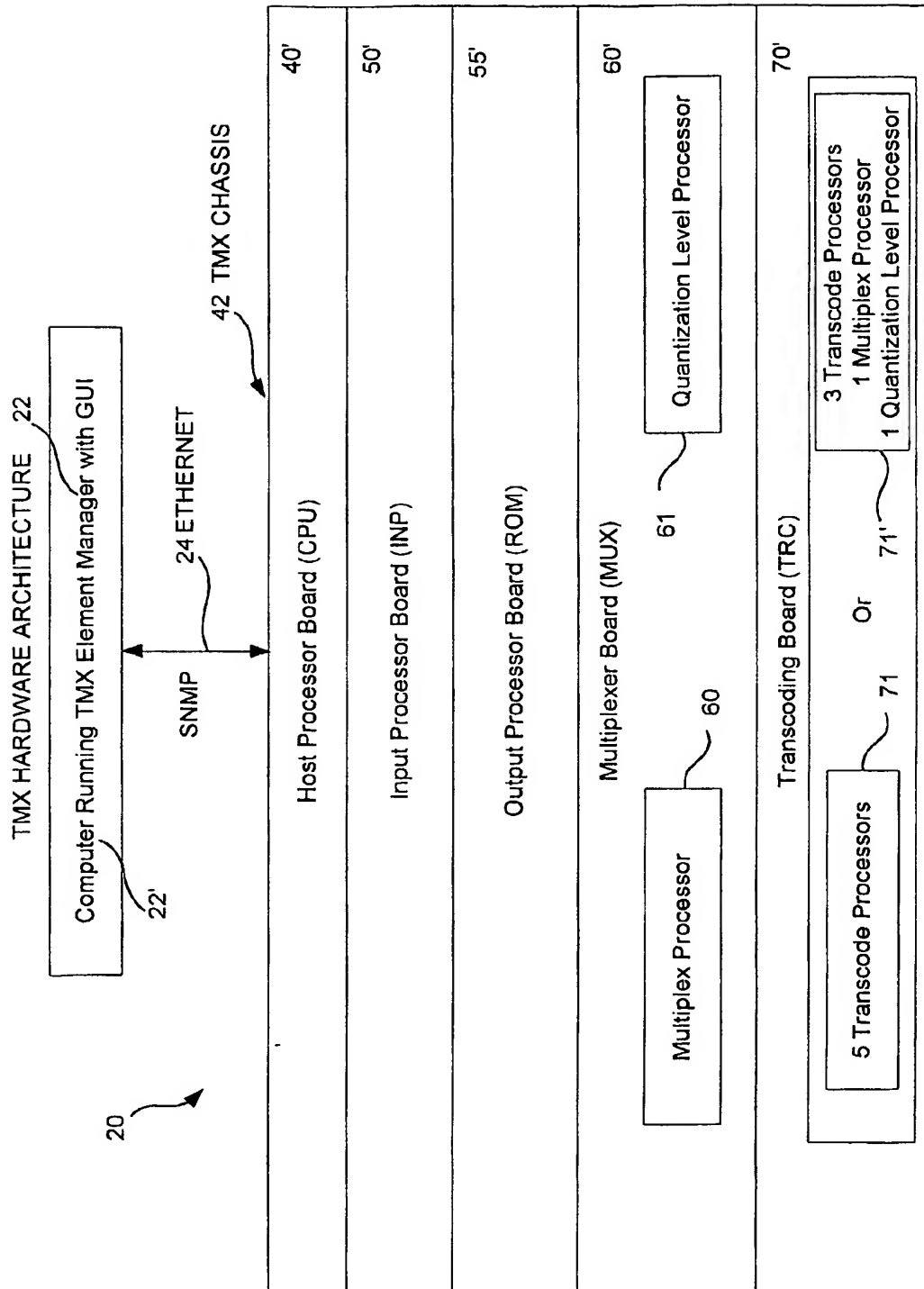


Figure 1a



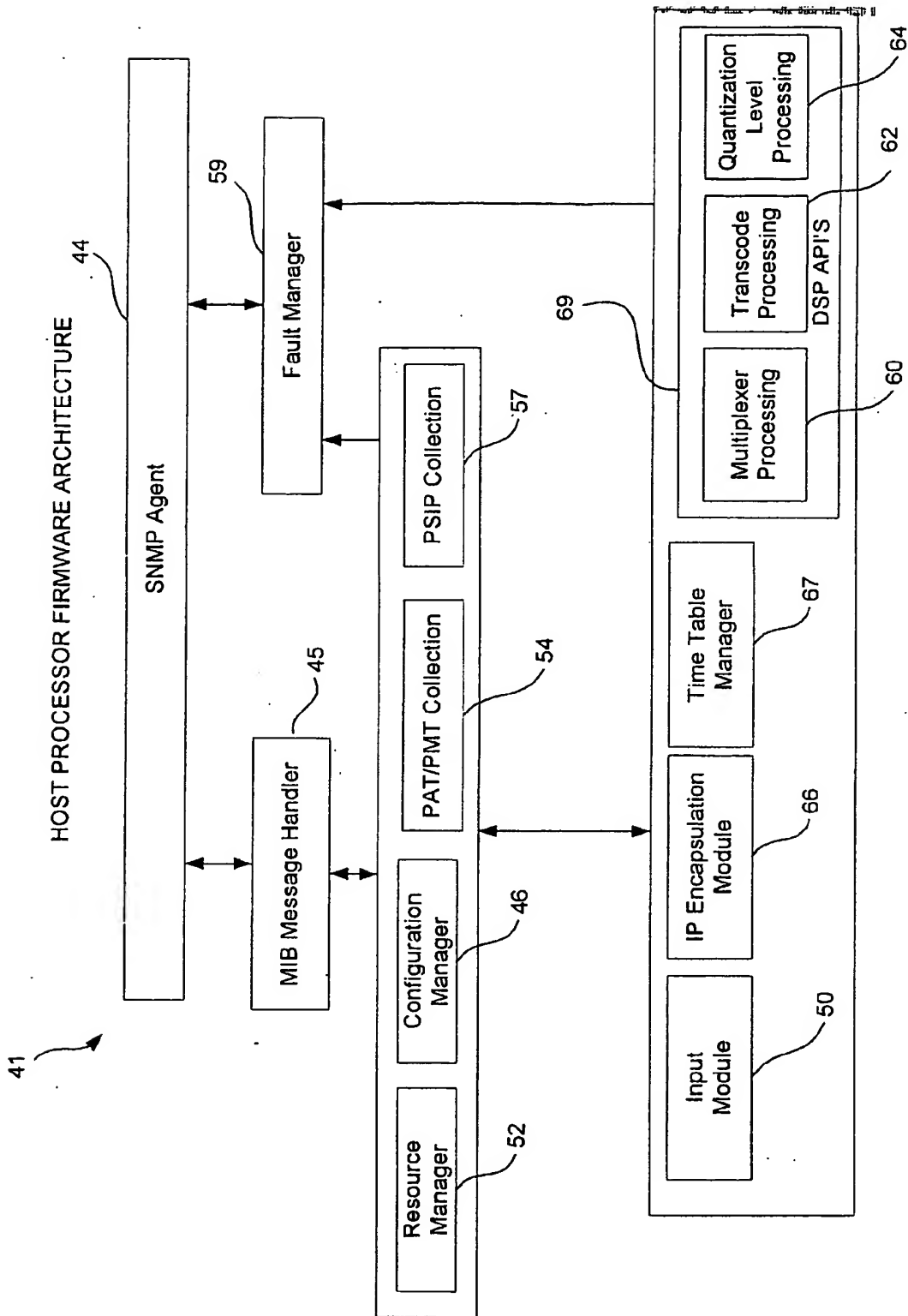


Figure 1b

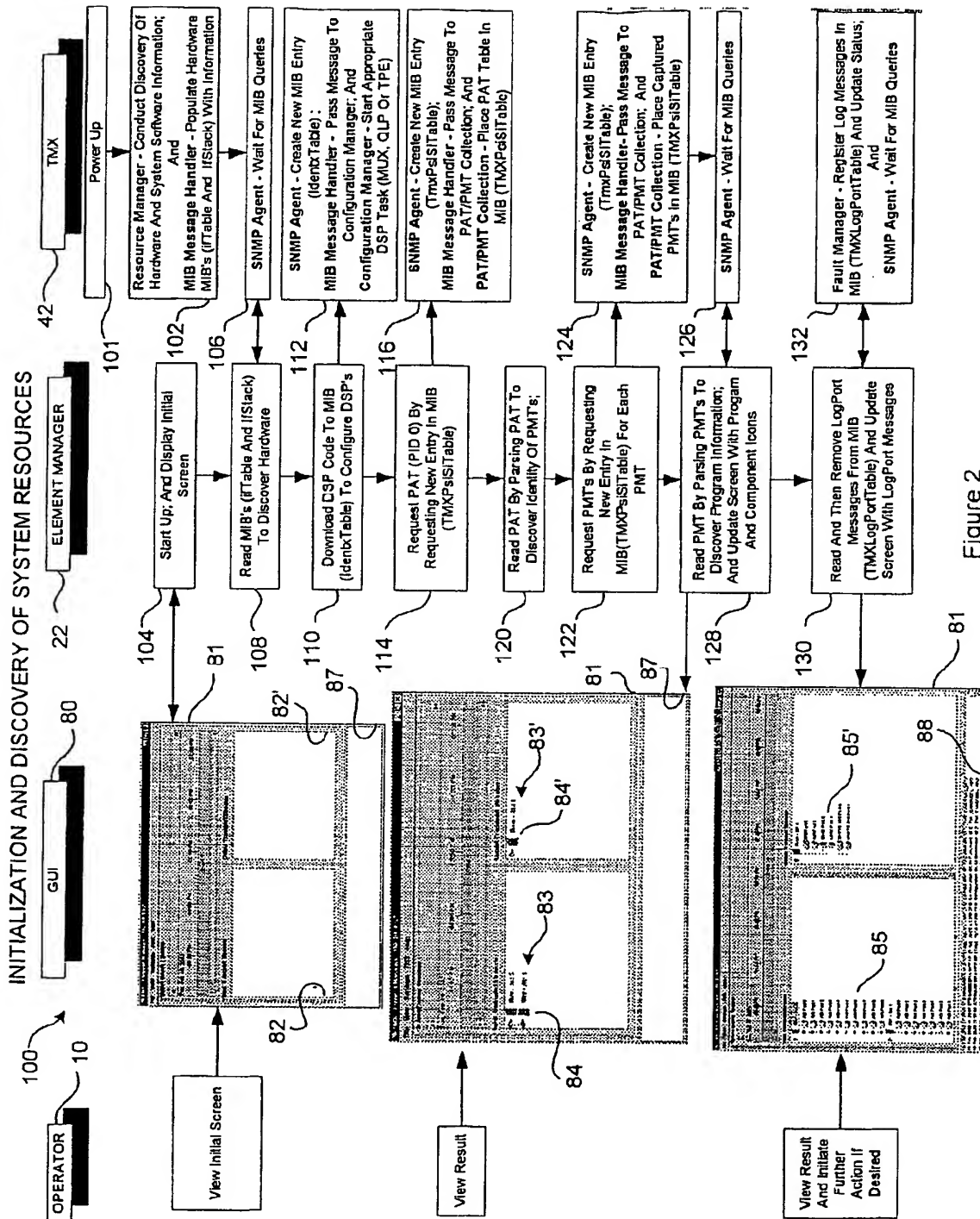


Figure 2

4 / 12

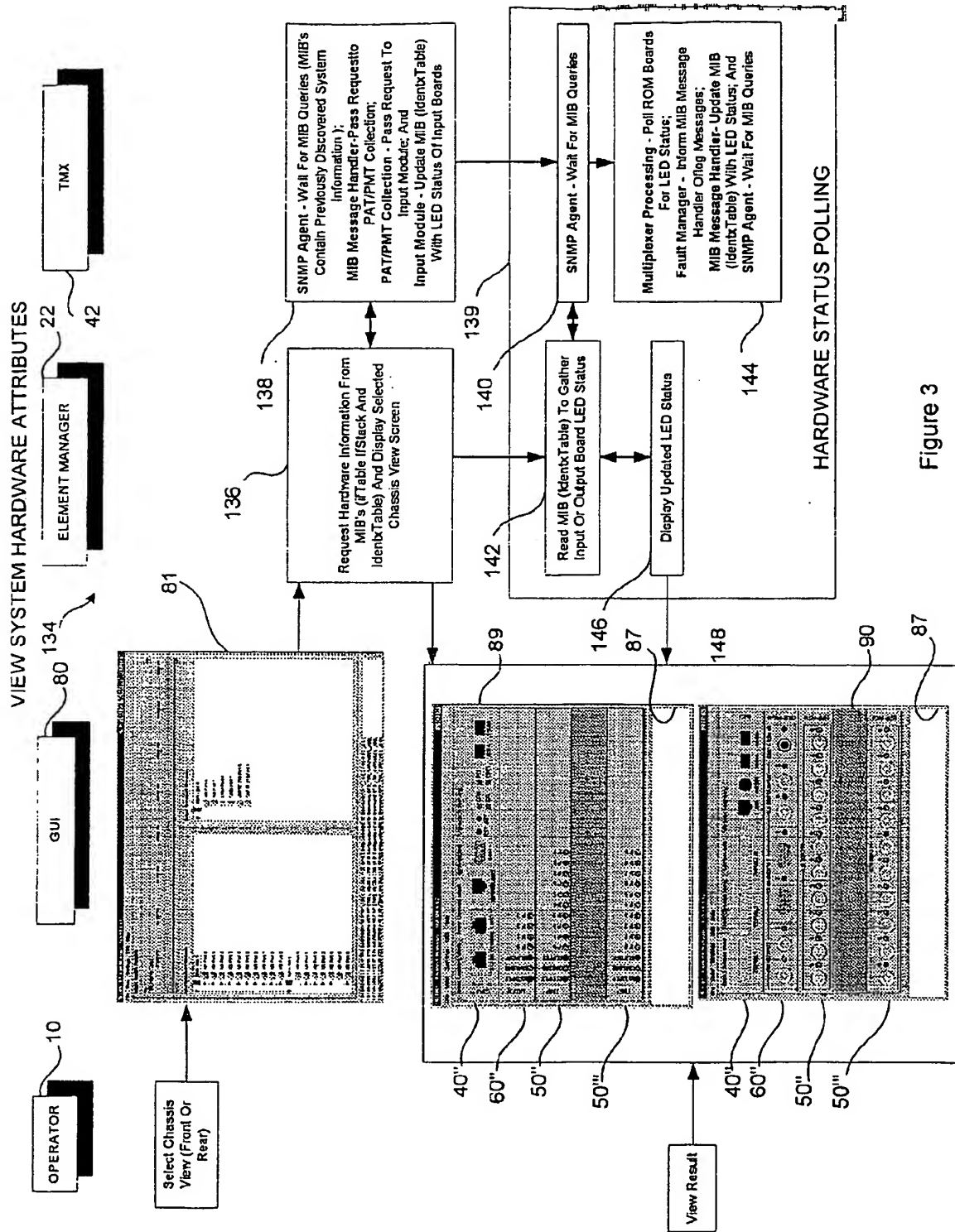
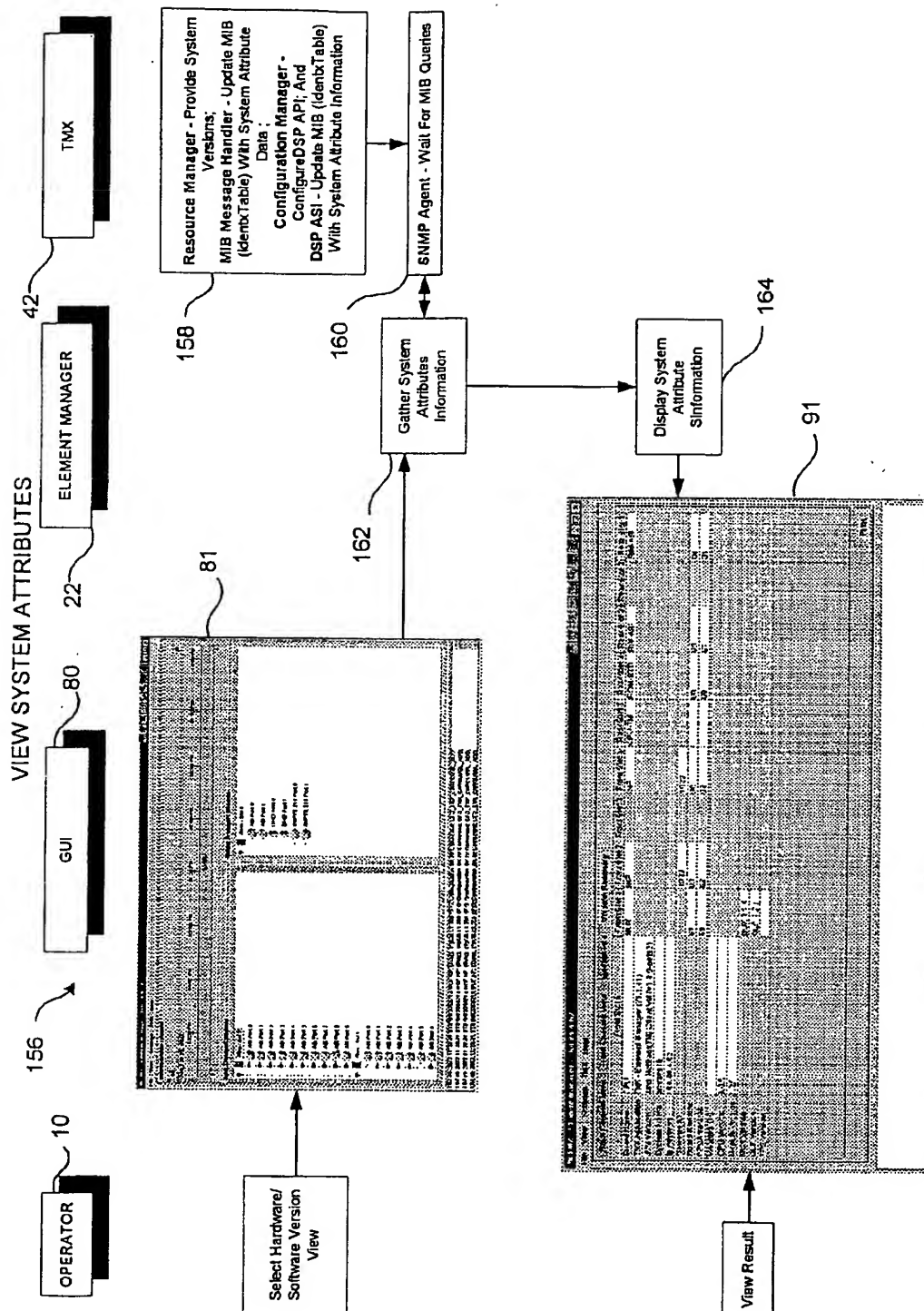
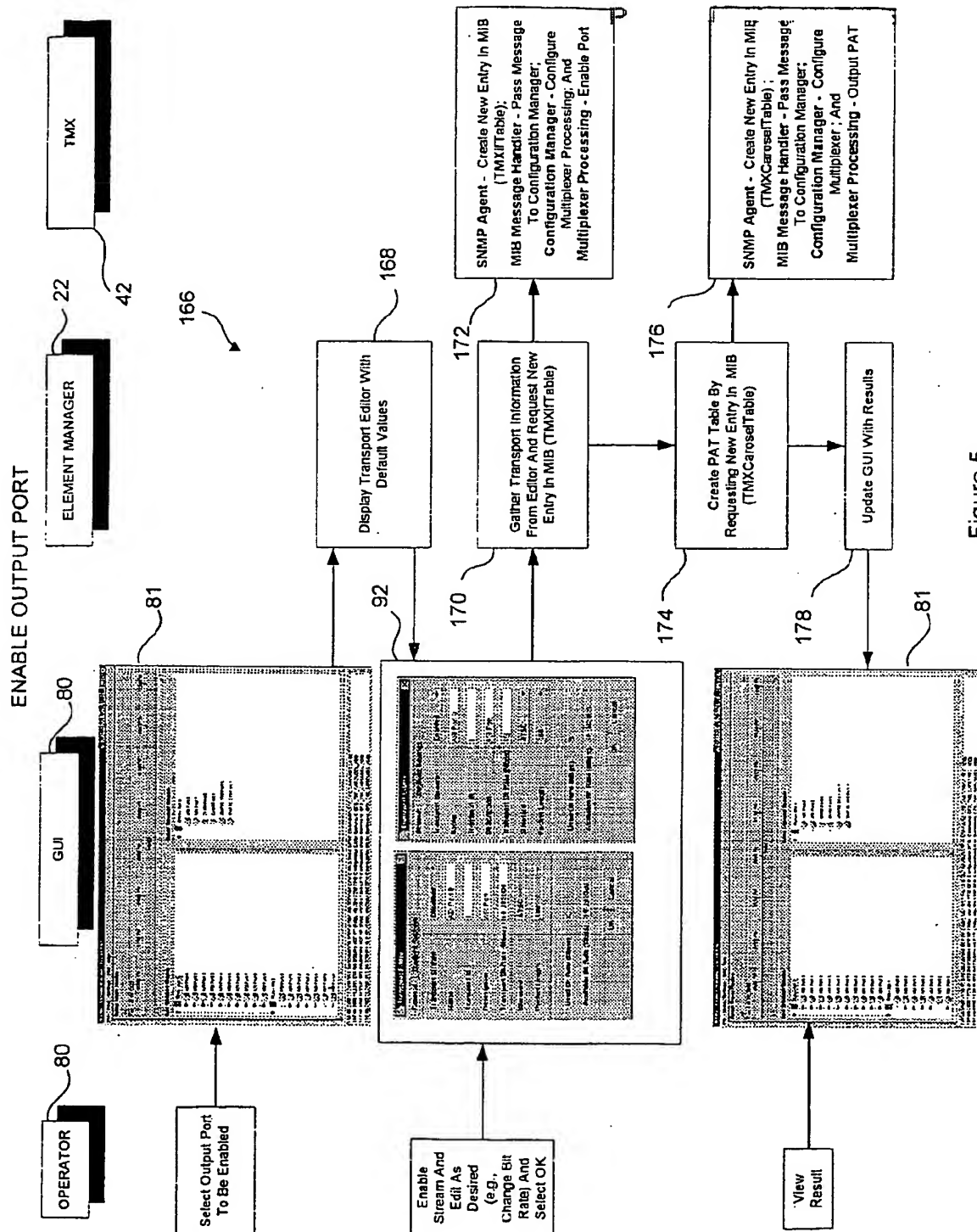


Figure 3

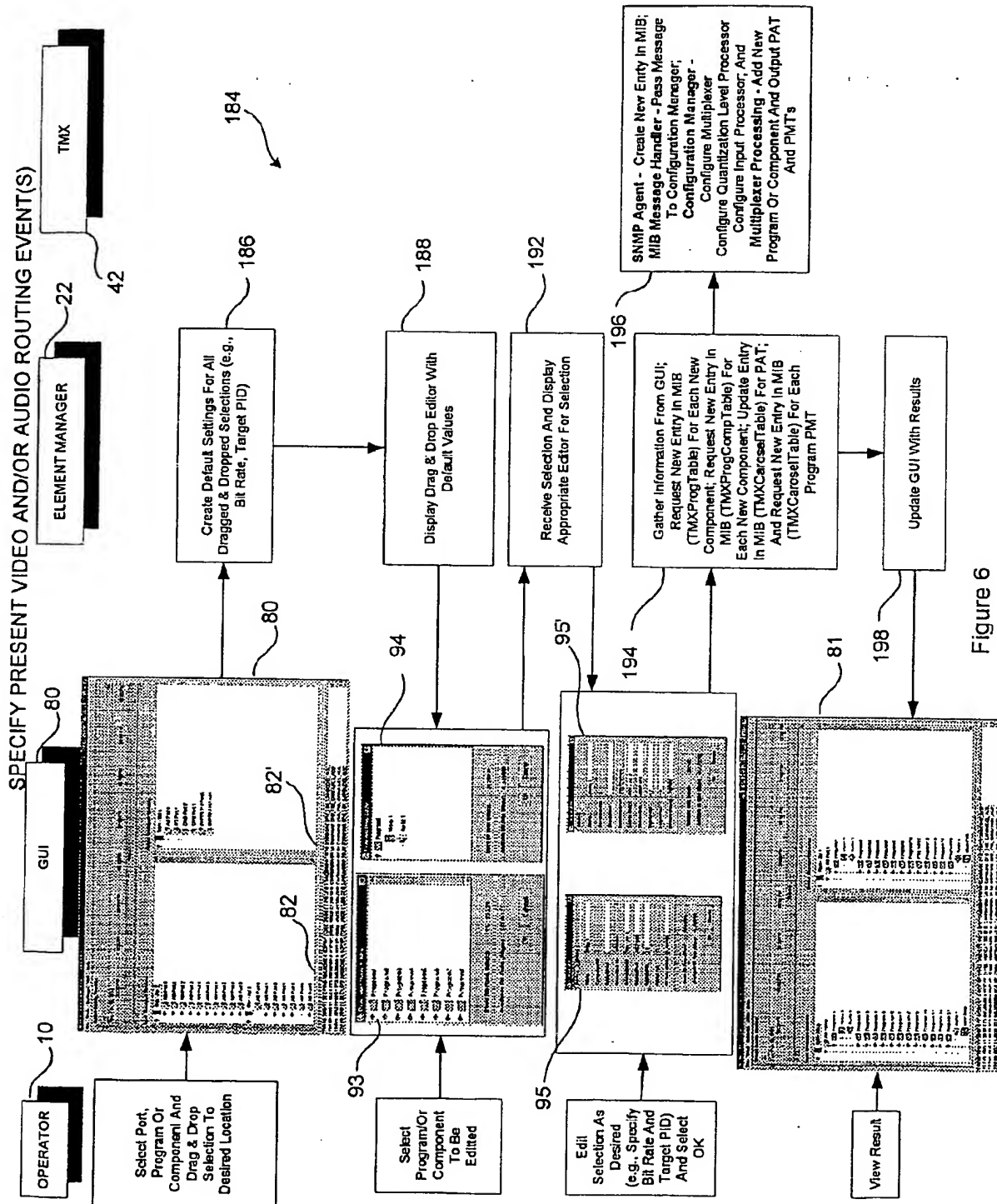


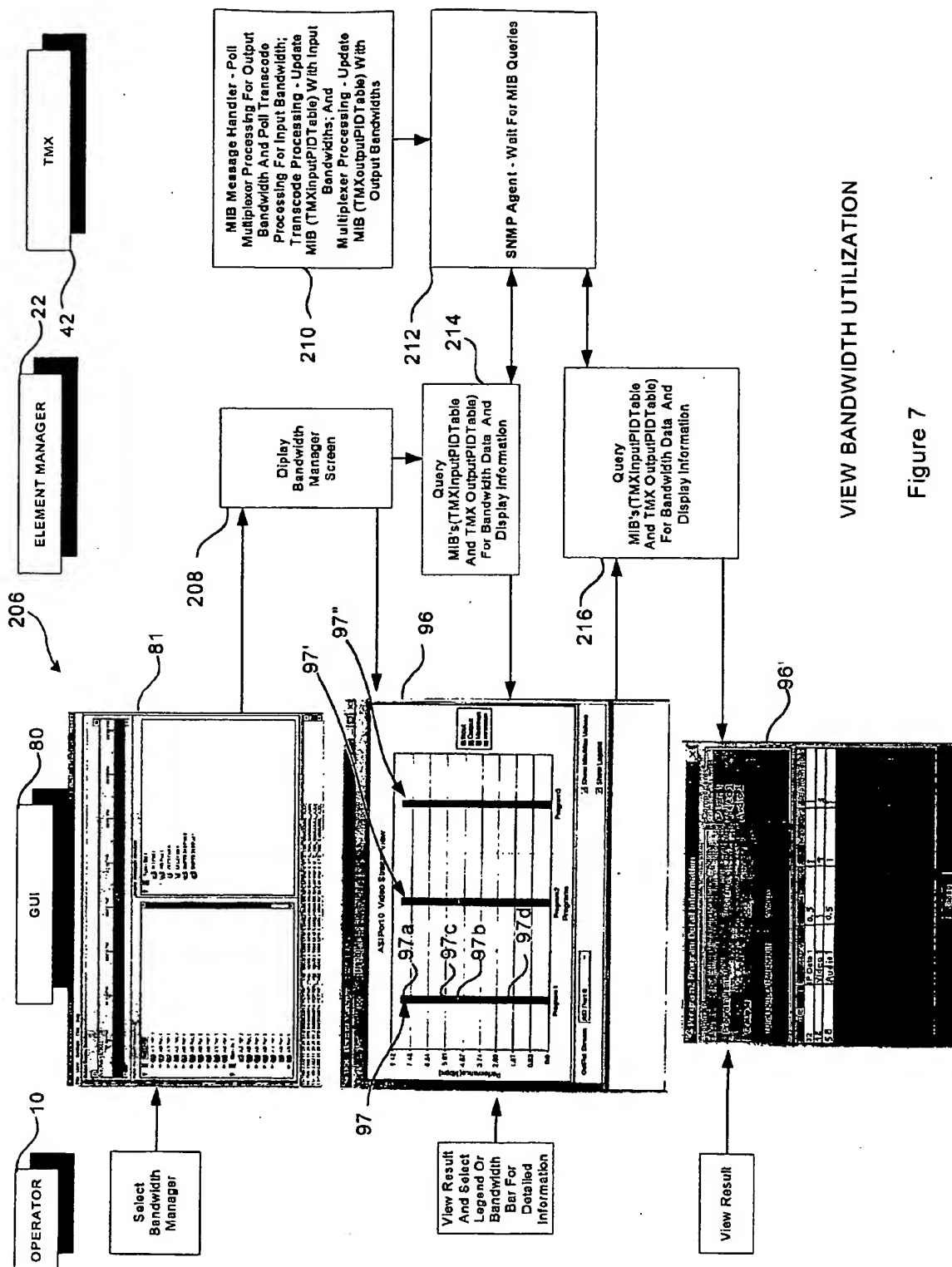
**Figure 4**

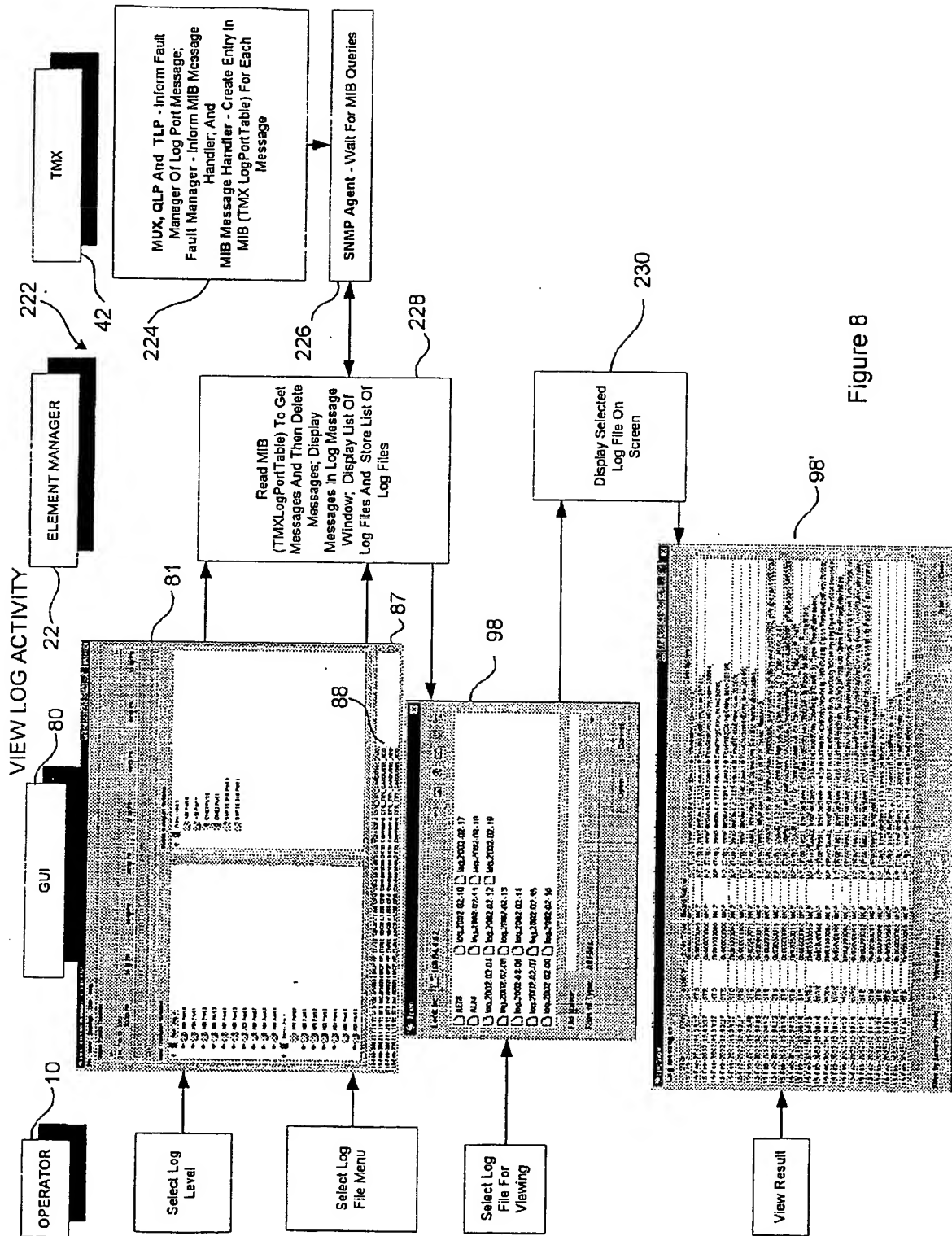


**Figure 5**

7 / 12









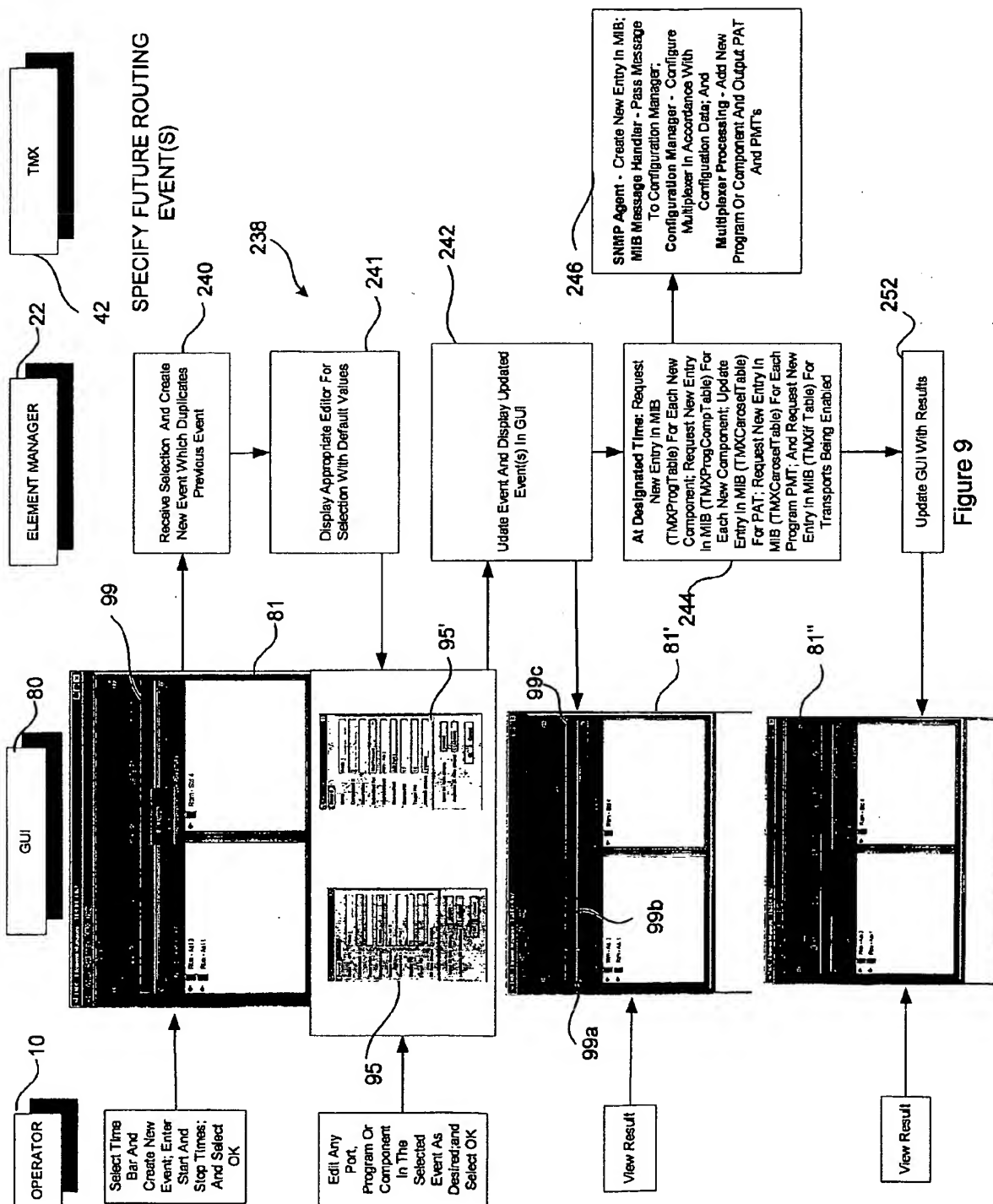


Figure 9

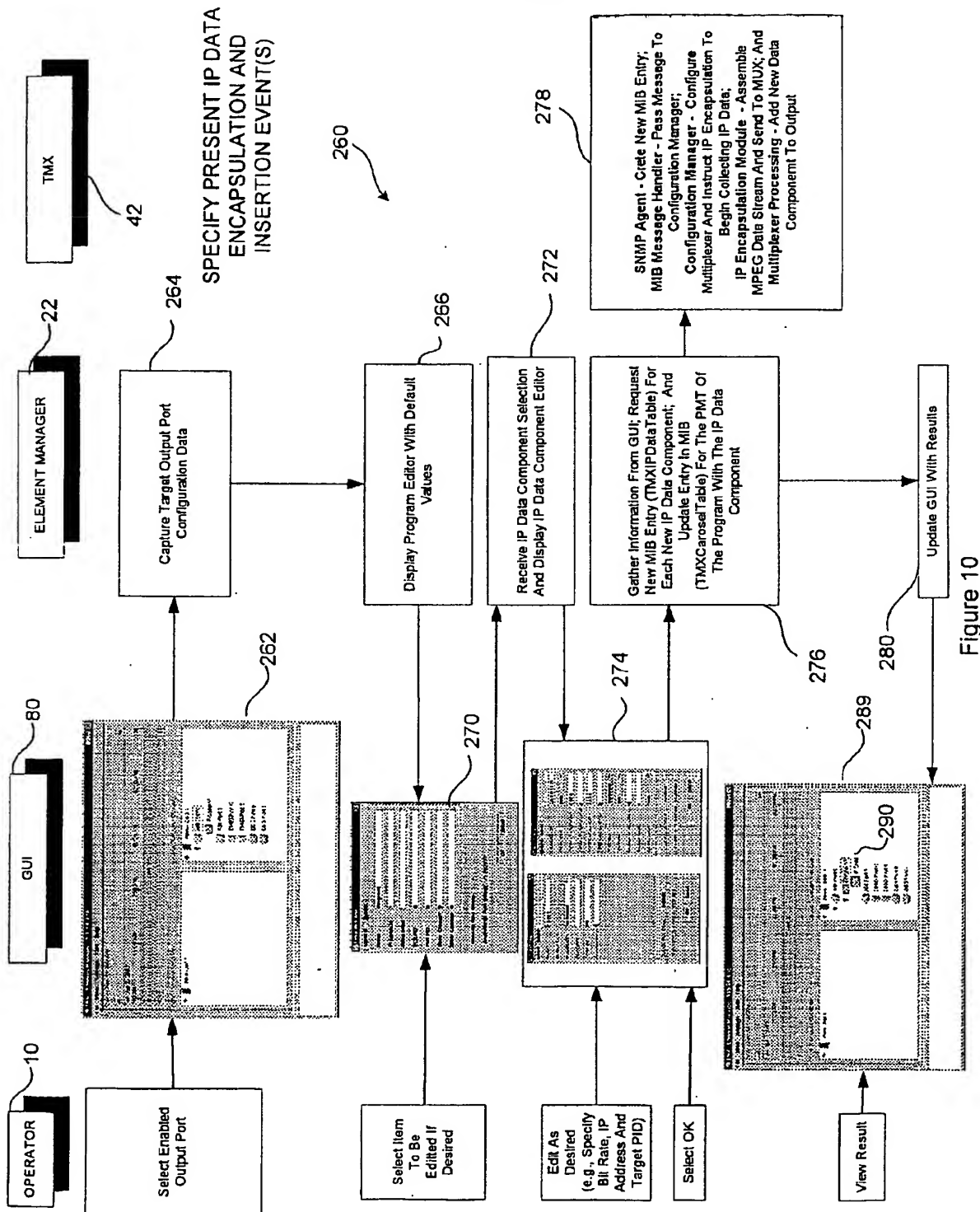


Figure 10

12 / 12

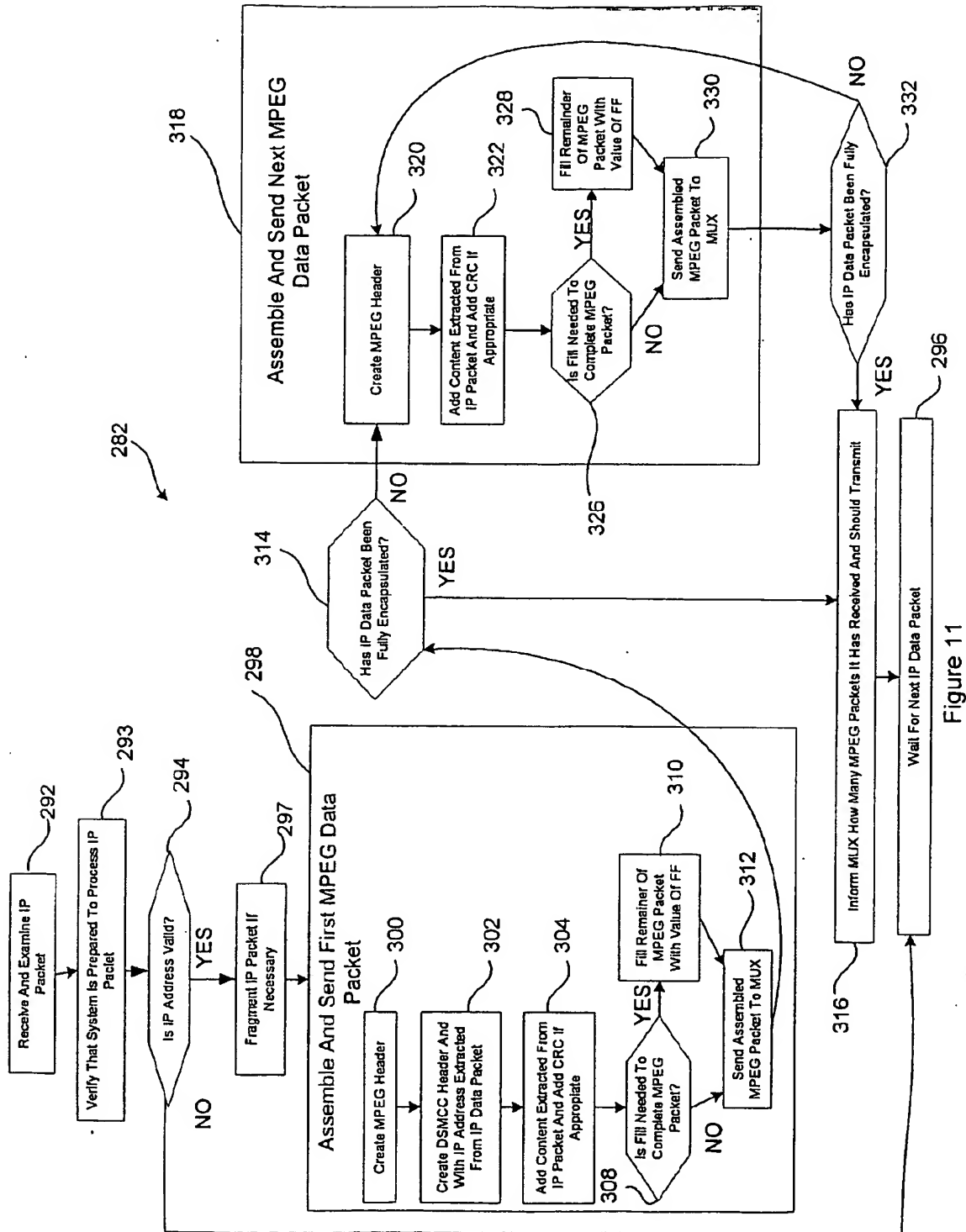


Figure 11

## INTERNATIONAL SEARCH REPORT

International Application No

PCT/US 02/12162

## A. CLASSIFICATION OF SUBJECT MATTER

IPC 7 H04L12/24 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 H04L H04J

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	US 5 825 775 A (LEE FRANK S ET AL) 20 October 1998 (1998-10-20)  abstract column 1, line 19 -column 5, line 4 figure 3  ----	1-4, 6-10,18 5,11,18
Y	US 6 182 249 B1 (CHU KEVIN L ET AL) 30 January 2001 (2001-01-30) abstract figure 7  ----- -/--	1-4,6,18



Further documents are listed in the continuation of box C.



Patent family members are listed in annex.

## \* Special categories of cited documents:

\*A\* document defining the general state of the art which is not considered to be of particular relevance

\*E\* earlier document but published on or after the International filing date

\*L\* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

\*O\* document referring to an oral disclosure, use, exhibition or other means

\*P\* document published prior to the International filing date but later than the priority date claimed

\*T\* later document published after the International filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

\*X\* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

\*Y\* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

\*g\* document member of the same patent family

Date of the actual completion of the International search

6 September 2002

Date of mailing of the international search report

24/09/2002

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Nocentini, I

## INTERNATIONAL SEARCH REPORT

International Application No

PCT/US 02/12162

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	<p>COMPUTER AUTOMATION SYSTEMS, INC.: "Integral Chassis 6U (10,5")" INDUSTRIAL COMPUTER, 'Online! 1999, pages 1-2, XP002212597 Retrieved from the Internet: &lt;URL:http://www.cas11.com/monitor6U.htm&gt; 'retrieved on 2002-09-05! the whole document -----</p>	7-10

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/US 02/12162

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 5825775	A	20-10-1998	NONE	
US 6182249	B1	30-01-2001	NONE	